3D INTERACTION STUDIES USING THE SHAPE-MATCHING PARADIGM

by

STANLEY JANG

B.Sc., The University of British Columbia, 1990

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October 1992

Abstract

At one time, 3D computer applications were only available on very specialized and very expensive workstations. However, technology on typical desktop computers has improved dramatically over the last decade, and is now capable of 3D graphics. If shrewd choices are made concerning the graphics hardware and software, a reasonably priced desktop computer will allow adequate real-time interaction for many applications. This thesis investigates some of these choices using a generalized program written to serve as an expandable testbed for evaluating some of the factors affecting user interaction in three dimensions.

An important set of 3D tasks that occurs in applications such as computer aided design and computer animation is curve or surface design. Not only is the 3D environment an essential issue in efficiently completing these tasks, but so are the properties of the spline formulation and spline interaction technique chosen to represent the curves or surfaces. Previous work has shown that the B-spline formulation is quite effective for the interactive construction of 2D spline curves. A possible drawback in using the B-spline for the more difficult task of 3D curve construction is the fact that its control vertices may lie far away from the curve, making its manipulation unintuitive. A direct manipulation technique, allowing a curve to be manipulated with points that lie on the curve itself, offers an alternative to control vertex manipulation.

An experiment was conducted using the testbed to compare the interactive design of 3D curves using control vertex manipulation of B-spline curves and a particular type of direct manipulation of B-spline curves. The results of the experiment revealed that “direct manipulation” was significantly faster than control vertex manipulation, without sacrificing accuracy in the shape of the final 3D curve.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Dr. Kellogg S. Booth: supervisor, Dr. David R. Forsey: faculty reader, Dr. Peter Graf (dept. of Psychology): faculty reader, experiment design and statistical analysis, Dr. Colin Ware: statistical analysis, Imager and GraFiC Laboratories through grants from NSERC and IBM.

Chapter 1

Introduction

With ever-improving computer technology fuelling the emergence of 3D interactive computer applications, there is a growing need to study and improve the interface between the human user and the computer. These changes often involve many compromises because although the needs and requirements of 2D computer interaction are well known, the needs and requirements of 3D interaction are still being discovered. For example, the mouse is an excellent two dimensional input device. The feedback obtained from the cursor on the screen makes its use very efficient and predictable. However, for 3D applications the mouse falls short by one dimension. Is the mouse sufficient for these tasks, or should users invest in one of the elaborate new 3D input devices now available? The mouse does prove to be adequate for many 3D applications if implemented with appropriate constraints.

In this thesis, an expandable testbed was written to study different interaction techniques for 3D applications. The program allows the user to select a variety of options affecting the interaction technique, the environment and the rendering used in the display during the execution of simple 3D tasks. A number of 3D tasks are available for testing. Some of the interaction techniques are constraints on the movement of points and the rotation of objects using the mouse. Options affecting the environment include the size of icons and windows, and mouse speed. Most of the display options are depth cues to help in visualizing 3D shapes. Using the testbed, 3D interfaces can be informally compared and new ideas may be implemented without excessive modifications to the program.

The testbed is also capable of conducting formal experiments to analyze 3D interaction. It contains software to provide a complete on-line tutorial, handle experiment administration and tabulate results, including a full record of subjects' actions. To test the software, a "shape-matching" experiment was conducted. Research used this paradigm for the study of 2D spline curves at the University of Waterloo. The experiment in this thesis is an extension of that work, but dealing with 3D spline curves. It investigated two techniques for 3D B-spline manipulation. B-splines are *approximating* splines, meaning that their control vertices do not in general lie on the curve. Compared to an *interpolating* spline in which control vertices do lie on the curve, control vertex manipulation of a B-spline may be less intuitive. It is not always obvious which part of the curve a given control vertex affects, especially for 3D curves represented by a 2D projection on the display screen. A different set of manipulation points can be created, all of which do lie on the curve. Using these points, the B-spline behaves much like an interpolating spline. 3D curve manipulation using these points may be more intuitive than using the control vertices. The 3D shape-matching experiment discussed in this thesis formally compares these two manipulation techniques.

The effort in modifying the source code to work with 3D splines is a good example of the difficulty in implementing a 3D application using an input device originally designed for 2D applications.

Although mathematically 3D splines are not much more complicated than 2D splines, the interface allowing the curves to be manipulated requires much work. The interface is more elaborate for two reasons. First, in 2D there are but two degrees of translation and only one degree of rotation, while in 3D there are three degrees of translation and three degrees of rotation. Second, the mouse and the CRT display, the standard input and output devices for a workstation, are both 2D devices. This hardware is fine for 2D spline manipulation, but not immediately adequate for 3D splines. The testbed can be used to explore interaction techniques using these devices for 3D tasks.

Interaction techniques for the generic 3D tasks that are used in the shape-matching paradigm are discussed in Chapter Two. The actions executed are split up into the three tasks: manipulating points, changing object orientation and rendering the 3D scene. Each of these tasks are discussed separately and typical examples for achieving them are given.

Chapter Three introduces the B-spline formulation. The shape-matching paradigm is defined and previous studies using it are reviewed. A direct manipulation technique for B-splines is introduced. Chapter Four deals with the direct manipulation experiment, explaining the experimental procedure, the selection of subjects, the environment, the results gathered, and a statistical analysis of those results.

The 3D testbed itself is discussed in Chapter Five, which describes all of the various options. Chapter Six summarizes the thesis and mentions some interesting future studies that are planned for the testbed.

Chapter 2

The Basic 3D Tasks

The majority of 3D applications for computer graphics workstations involve designing, manipulating or studying “real” objects. Examples include CAD (computer aided design), scientific visualization and computer animation. Some specific 3D tasks performed in these applications are the design of a house, the manipulation of models of chemical molecules or the study of a patient’s jaw in dentistry. Upon closer inspection, these tasks may be broken down into three basic, generic tasks that are executed over and over again. These basic tasks form the building blocks of the larger tasks. The basic tasks are:

1. manipulating points,
2. changing the orientation of objects relative to the viewer, and
3. rendering the 3D scene.

To illustrate the function of these basic tasks, the design of a house using a CAD system might begin with the creation and manipulation of a set of points to form the first wall. Successive walls would be constructed in the same manner, but the orientation of the partially completed house on the computer display would have to be changed to construct some of the walls. During this entire process, the 3D shape of the house as represented on the 2D display would have to be interpreted through rendering algorithms that might include perspective and hidden surface removal.

There are numerous techniques that are available for each of the basic tasks. No one technique is best for all applications, but some are geared more towards a specific type of application than others. Many of the techniques that will be discussed have been implemented in the testbed software.

Some of the discussion that follows deals with rotations of objects represented on the CRT display. The coordinate system used in these discussions will be as shown in Figure 2.1. The *x* axis runs horizontally across the screen, the *y* axis runs vertically along the screen, and the *z* axis runs out of the screen. This is the standard right-handed coordinate system.

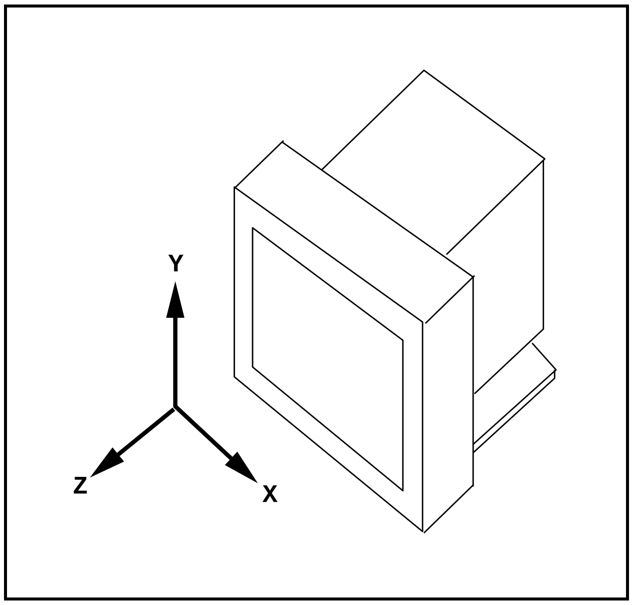


Fig 2.1 Right-handed coordinate system for objects represented on the display.

2.1 Manipulating Points

2.1.1 Selecting Points

The mouse, after the keyboard, is by far the most popular input device for computer graphics workstations. The mouse works well with the CRT display because one is a 2D input device and the other a 2D output device. Traditionally, many applications have been 2D. But for 3D applications, the mouse and the CRT each fall one dimension short. We can either use the mouse to provide constrained 3D interaction, or we can use a 3D input device. Constrained interaction is often a viable approach. In fact, the vast majority of 3D applications use the mouse as the primary input pointing device.

The most common technique is to allow the mouse to act as a 2D pointing device with a perspective or orthogonal projection on the display. Points can be selected by simply placing the cursor over the desired point. If a new point is being entered, the depth can be set to some default value[1]. If we are selecting an existing object and the particular view is cluttered with selectable objects, making it difficult to select the correct object, the program can simply pick the nearest object. This is the technique used for the shape-matching experiment presented later. Because the number of selectable points is very small, conflicts are rare. When a conflict does arise, the user can change the viewpoint or orientation of the scene so that there is no longer a conflict. If the scene is very complicated, this solution may not work because every view may contain a conflict. Many of the commercial 3D modelling systems simply pick all objects that are under the cursor. To overcome this, one can either pick from a different view, pick

[1]Methods for changing the depth are discussed in the next few sections.

the object by name, pick its icon from a list, zoom in until there is no longer a conflict, or set some of the obstructing objects to be invisible.

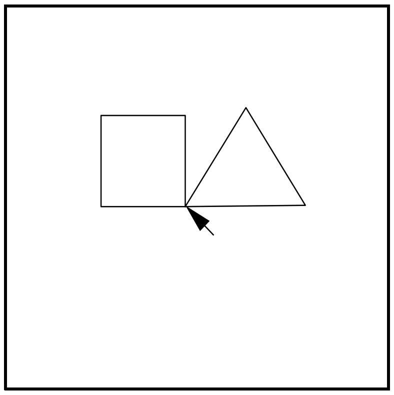


Fig 2.2a Pick vertex of square.

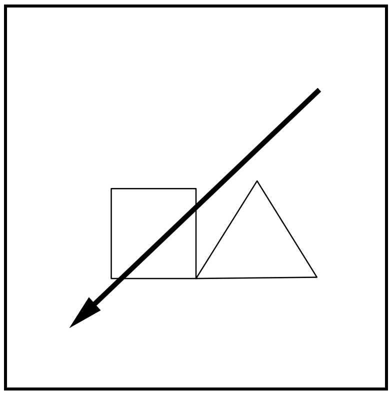


Fig 2.2b Rotate objects first.

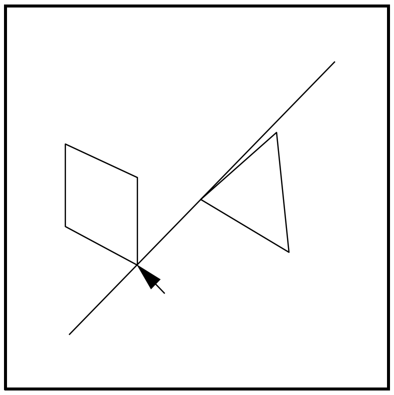


Fig 2.2c Now pick vertex.

An alternative technique, illustrated in Figures 2.2(abc), allows the user to input or select a 3D point without ambiguity, but requires two mouse button clicks. In the figures, the user wants to select the vertex of the square that coincides with the triangle. As before, the cursor is moved over the 2D projection of the object and it is selected (Figure 2.2a). The selection is ambiguous because the cursor is pointing at several objects. What the user has done is not pick a point in 3D, but a line in 3D that is represented as a point in the given projection. The set of possible objects lie on this line. To pick the desired object from the line of objects, the user can change the orientation of the scene (Figure 2.2b) so that the projection of the line is in fact a line. The user can then pick the object from the line of objects (Figure 2.2c). Essentially, the mouse is being used to generate two intersecting rays in 3D that will uniquely locate the desired point, where the final selection depends on two 2D point selections, an intermediate rotation of the objects with respect to the viewpoint, and the rendering of the auxiliary line determined by the first 2D point selection.

2.1.2 Moving Points

After a point has been selected, one may want to move or drag it to a different location. Again, there is the problem of the mouse not providing enough degrees of freedom. The most common technique is to allow the mouse to manipulate the point on a 2D projection of the scene while holding the depth constant (i.e., movement is constrained to a plane parallel to the screen). Given the ability to work from a different view, the point can then be moved in all three dimensions. Movement may also be constrained to some oblique plane or to a line in 3D for a specific application. For example, some commercial modellers allow constrained movement horizontally or vertically on the screen.

Constrained motion using a mouse may sound more restrictive than it really is. On a CRT display, even with the best depth cues, it is difficult to accurately modify the depth of an object (i.e., into or out of the screen). So, to change the depth of a point, one would likely prefer to work with an orthogonal view where the corresponding movement is parallel to the screen, even if unconstrained 3D movement was available from the input device.

2.1.3 Other Input Devices

As mentioned earlier, the mouse is by far the most common input device used for pointing. In fact, every current workstation comes equipped with one. The mouse has proven to be extremely versatile for the variety of software applications available. However, some complicated tasks benefit from an input device with more degrees of freedom. These devices are not very popular for a number of reasons including cost, hardware compatibility, software compatibility, and ease of use or effectiveness. They are used primarily in research labs and for some special applications. Most of these input devices offer six degrees of freedom, three for translation and three for rotation. Below is a partial list of current six-degree-of-freedom input devices.

Polhemus Isotrak

The Isotrak uses a pulsating magnetic field to determine position and orientation. It consists of two components, a source and a sensor. The source remains fixed and emits the magnetic field, while the sensor, a very small unit, may be placed around a finger or mounted on one's head. Both the source and the sensor contain three mutually orthogonal coils. The coils in the source are pulsed at separate times, and the coils in the sensor record the magnitude of each pulse, providing nine values that are used to determine the position and orientation of the sensor with respect to the source. The Polhemus offers an adequate sampling rate and fairly good accuracy, but is susceptible to noise from other electromagnetic devices and metallic objects, and it is expensive ($3000).

Ascencion Bird

The Bird is similar to the Isotrak except that it uses electromagnetic radiation instead of a pulsating magnetic field. In comparison, it provides a higher sampling rate and is less susceptible to interference.

Logitech 6D Mouse

Unlike the two previous devices, the Logitech mouse uses ultrasound between the transmitter and receiver. The transmitter is a triangle with speakers at each of its corners. The receiver is the 6D mouse, which contains three microphones to sample the signals from the transmitting triangle. The mouse also operates in a 2D mode, similar to a cordless 2D mouse. Compared to the Bird and the Isotrak, it is less sensitive to disturbances from surrounding objects (unless they emit ultrasound) and is much cheaper ($1000), but it lacks the same degree of accuracy.

Spaceball

The Spaceball is a six degree of freedom joystick. It is not as versatile as the other devices in that it cannot be mounted on some part of the body and moved freely in 3D. The user grasps a large ball at the end of the joystick that can be moved forward-backward, left-right, and up-down, as well as rotating it about its three degrees of rotation (pitch, yaw, and roll). Unlike a joystick, the “stick” does not move. Inside the large ball that one grasps, is a smaller ball which is fixed to the stick, and thus does not move. These two balls are attached to each other by a number of springs. So, the large ball is free to move relative to the smaller ball inside. It is the force on these springs that are used to calculate the six degrees of freedom. The most common complaint about this device is the lack of tactile feedback provided by the device itself. The large ball of the joystick moves very little because it is the force on the springs that are measured, not a displacement in translation or rotation. Because the movement is minimal, the user does not know if the correct input has been applied unless the application program provides adequate visual feedback on the display.

ADL-1 6D Tracker

The 6D Tracker employs a mechanical arm to determine position and orientation. It consists of the main unit, a mechanical arm with two limbs, and a harness for one's head. The main unit is fixed and contains a microprocessor for calculating the six parameters. The arm connects the main unit to the harness. The harness can either be secured to one's head, or it can be replaced with a handle for hand manipulation and used as a pointing device. There are a total of six precision potentiometers at the joints of the 6D Tracker that provide the six degrees of freedom. Compared to the Isotrak or the Bird, this device is more burdensome when mounted on one's head. But, because it uses simpler technology, it is available for about half the cost and is not prone to interference. It is capable of sampling rates at least as high as the other devices.

2.2 Modifying An Object's Orientation

The previous section has revealed the importance of having different orientations of the scene or object for a 3D task. Even in the real world, we rotate objects in our hand to study them. The view of the objects in a scene may be modified in two ways: either by rotating the object in question about its centre[2], or by changing the eyepoint of the viewer in the scene. The latter requires control of all six degrees of freedom whereas the former only deals with the three rotation parameters. In both cases, the mouse lacks the required degrees of freedom.

[2]Any one of the calculated centres will suffice.

Modifying the eyepoint is commonly done when the displayed projection represents a camera, as in 3D modelling/animation software, or the particular hardware includes a head-mounted tracking device such as those mentioned previously. Because there are six degrees of freedom when manipulating the eyepoint, doing so without a 6D input device (i.e., using a mouse) is very tedious. For example, to modify the eyepoint for the perspective window in some commercial modelling software, one must first select the parameter to be changed and then change that parameter using the mouse or keyboard. It typically requires several changes to obtain the desired view.

To rotate an object, a centre of rotation must be defined. This point should be close to the centre of the object, otherwise the object will orbit around the point and fly off the display screen when rotated. One can imagine that the ultimate input device for this task would be a model of the object which the user could hold in his hand and orient as desired, causing the object represented on the screen to rotate accordingly. One can then generalize this idea by having a sphere or some other simple object that one can hold in his hand as the input device. Many of the input devices described in the previous section are used in this way to modify the viewing rotation parameters.

Because of the constraints in using a mouse for this purpose, much research has been performed to maximize its intuitiveness and efficiency. With a bit of software, the mouse can be used as a “virtual trackball”, imitating trackballs that may be found in video games and as input devices for some computers. This interface for the mouse maps the three degrees of object rotation to the two degrees of mouse motion in a predictable manner. For example, a very simple implementation is to let horizontal movement of the mouse correspond to rotation about the *y* axis and vertical movement correspond to rotation about the *x* axis. Rotation about the *z* axis is never performed. This implementation is fine for gross movements, but lacks intuitiveness when attempting accurate, fine rotations.

A better approach is one published by Chen, et al. [Chen90]. Their approach simulates a real trackball input device. The imaginary trackball, or sphere, surrounds the object on the display screen. The mouse or cursor can then be used to rotate the object by rolling the imaginary sphere. When the mouse is moved, a vector is generated which in turn generates an axis of rotation and an angular velocity. In general, the rotation generated is a product of rotations about all three axes. Pure x or y rotations are obtained by moving the mouse along the horizontal or vertical axis of the imaginary sphere, and pure z rotation is accomplished by moving the mouse around the circumference of the sphere. Shoemake and others have devised variations on the virtual trackball [Shoe92].

Given one's favourite virtual trackball implementation, there are still a number of different techniques for interaction. The techniques described thus far are *positional* techniques where each incremental movement of the mouse leads to an incremental rotation of the object. If the mouse does not move, the object does not rotate. For these techniques, the only mouse information required is the current location of the mouse and its previous location. Another class of interaction techniques, called a *directional* trackball, looks at the original position of the mouse (i.e., the location of the mouse when the virtual trackball was invoked) and the current position. These two points define a vector whose direction may be used with a constant angular velocity to generate incremental mouse locations to be given to the positional trackball to rotate the object. This is not another virtual trackball implementation, merely a different interaction technique. Because only the original and current mouse positions are used, the object is constantly rotating unless these two mouse positions coincide. If the mouse is held still, rotation continues about the same axis. Essentially, the user is able to perform large rotations smoothly without having to pick up the mouse. The directional trackball is useful for the visualization of a complicated object where smooth continuous rotation helps in studying its 3D shape. With the positional trackball, this kind of rotation is just not possible. From personal experience, the disadvantage of this interface over the original is that finding a precise view is difficult. This problem is significant during construction or editing tasks where one may want a particular view of the object.

An extension to the *directional* trackball interface looks at the magnitude of the vector formed by the original and current mouse positions as well as the direction. The magnitude can be used to set the rate of the object's rotation, as opposed to keeping a constant rate of rotation. This *rate-controlled* trackball is easily implemented given the directional trackball. The behaviour of this interface is similar to the directional interface except that the user has more control. By keeping the mouse close to the original position, the user can rotate as slowly as desired, allowing more precision. However, it is difficult to make small adjustments in the direction of the vector and thus the direction of rotation when the vector is so short.

2.3 Rendering the 3D Scene

The CRT display is a two-degree-of-freedom output device. When the scene contains 3D data, the user is once again constrained. Essentially, what is rendered is a 2D projection of the scene. This may appear to be sufficient because pictures and photographs are 2D projections and they do a good job of portraying the 3D world. But photorealistically rendered objects are very compute intensive, so often the user is shown only a wire-frame representation or a polygonal approximation. During the construction of a 3D object (i.e., in CAD applications), the user must have more than a 2D projection. Depth information is needed as well. Numerous techniques for providing depth cues on a computer display are available. Some of these are discussed below.

Multiple simultaneous views

The simplest method is to provide multiple views. Architects and draftsmen used this technique well before the advent of computers. Many CAD software and 3D modelling/animation systems incorporate this technique. Usually, three orthogonal views and possibly another arbitrary view are provided. For example, these may include the front view, top view, right view, and the camera's view. The first three views are usually used for the construction of the 3D object, and are orthogonal projections, while the camera's view, a perspective projection, is used for visualization of the object. It is difficult to build objects in a perspective projection, so construction usually takes place in the three orthogonal windows.

Perspective projection

Normal visualization of 3D objects relies on a perspective projection rather than an orthogonal projection. The geometry of objects in a perspective projection is the same as that in images formed by the human vision system, so the scene appears more natural. With this projection, distant objects are made smaller than near objects and parallel lines travelling away from the viewer converge in the distance. Sometimes the geometry is exaggerated slightly to strengthen its effect as a depth cue. Other depth cues may be used in conjunction with a perspective projection.

Z-buffer Hidden Surface Removal

Typically, a computer displays every object in the scene in some systematic order regardless of the location of the objects in relation to each other (i.e., objects blocked from view are drawn as well). If the objects are wireframe, one sees all of the lines, but if the objects are shaded, the projection can be quite confusing. An excellent depth cue is to remove portions of objects that are occluded in the given view. Hidden surface removal is primarily used for visualization and not for construction because information about occluded objects is lost when the depth cue is used. A number of algorithms exist for this task, but it is frequently performed in hardware with a z-buffer. The pixel location of every point that must be drawn is computed. These pixel values (RGB or colour table value) are then stored in the frame buffer while the depth of the point represented by each pixel is stored in the z-buffer. The z-buffer contains storage for the depth of every pixel on the display. If more than one point is drawn on the same pixel, their depth values are compared and the one closer to the viewer is retained in the frame buffer. When every point of every object has been computed and all have had their depth values checked, the pixel data stored in the frame buffer is displayed on the screen.

2D Grid

A 2D grid oriented in the *x*-*z* plane with the lines running parallel to the axes placed in a perspective projection provides the user with an inexpensive depth cue. Because of the projection, the lines of the grid parallel to the *z* axis converge in the distance. Some commercial 3D modelling/animation systems use this grid in the camera’s window.

Mesh surface

Many 3D models can be effectively represented on the computer display with just a wireframe. But what if the model consists primarily of curved surfaces? There are no edges to display and simply drawing the outline is not adequate. A mesh representation of the surface can deliver the necessary shape or curvature information. Used in conjunction with hidden surface removal, the user is given a very useful representation of the object.

Flat or Interpolated Shading

A shaded image of a scene is visually more pleasing than a wireframe representation. All objects in the real world have some “colour” associated with them. Flat shading is the simplest form of shading, covering each polygon representing the object with a uniform colour. However, most objects, especially those with curved surfaces, vary in their colour because of light and surface properties. Interpolated shading can be used to blend the colours between a number of points. For a triangle, one can assign different shades to each point and interpolated shading will produce a triangle with smooth variations in colour. Varying the colour of a curved surface in this way approximates the smooth changes in its shading. Gouraud shading is a kind of interpolated shading that minimizes intensity discontinuities.

Shadows

Shadows cast by objects provide further realism and provide an additional depth cue. A shadow gives some indication of the distance of objects from the light source and the relationship between two objects if one object's shadow hits the other object. Rendering a scene with light sources, however, is very expensive. Some research has been done with fake shadows to determine their effectiveness as a depth cue [Wang92]. These fake shadows are in the correct location, but they are of a simple shape such as a circle or a square.

Intensity Cue or *Z* axis Modulation

Another depth cue is to set the colour of an object depending on its relative depth. In general, objects farther away from the viewer approach the background colour while objects closer to the viewer approach some chosen colour for the object. This technique is inexpensive and very effective.

Kinetic Depth Effect

Relative depths between objects may be determined when there is motion involved. Consider the case when the motion is rotation about a vertical axis inside an object. For points that lie between the viewer and the rotation axis, the closer the point is to the viewer, the faster it moves. In order for this depth cue to be effective, the hardware must be able to produce smooth motion. Several years ago, this would have been much to ask for, but today, even desktop personal computers have the necessary compute power.

Stereopsis:

A depth cue that the human vision system utilizes when viewing a 3D scene is binocular disparity. It relies on the fact that humans have two eyes that have similar but slightly different points of view. Each eye receives a 2D image of the 3D scene. The two images differ in that some objects present in both images are not in the same horizontal location. This difference in the horizontal location is called *horizontal disparity*. The amount of horizontal disparity depends on the depth of object in the view. Objects far away have very little horizontal disparity while objects very close to the viewer have much more.

Binocular disparity does not exist when viewing a picture or a 2D image. Each eye receives the same image so the horizontal disparity is zero for all objects. To obtain this depth cue when looking at a 2D representation of a 3D scene, each eye much be given a different image. On a computer workstation, this may be accomplished with the following techniques: polarizing the light from the two images in different directions and having the viewer wear glasses with polarized lenses so that each eye sees one image but not the other; using colour filters instead of light polarization: or flashing the two images alternately on the same monitor and having the viewer wear glasses that have electronic “shutters” opening and closing in synchrony with monitor. To obtain smooth motion, the monitor should have a refresh rate of at least 120 Hertz. Another solution is to have the user look at two monitors, one for each eye. Some head-mounted displays used for virtual reality applications employ this technique.

Chapter 3

Spline Curves and the Shape-Matching Paradigm

In this chapter, some background information is presented on splines, in particular B-splines, to prepare the reader for the discussion of the curve-matching experiment discussed in Chapter Four. The shape-matching paradigm is described and results from previous shape-matching experiments are summarized, revealing the reasons for choosing B-splines in this experiment. A comparison between 2D and 3D spline curves is given to justify the hypothesis that curve manipulation in 3D is more difficult than manipulation in 2D. A "direct manipulation" technique for B-splines is introduced to alleviate this problem.

A spline is a class of mathematical formulations for representing curves. A spline curve is made up of a number of *curve segments* each of which is represented by a set of polynomials (often cubic). The coefficients of the polynomials are determined by a basis matrix (or a set of blending functions) and a vector of geometric constraints (called a geometry vector). These constraints are a number of 3D points called *control vertices*. The blending functions are unique for a given spline formulation. So the equation for a curve segment $Q(t)$ is $Q(t) = T \cdot M \cdot G$ where T is a vector of polynomials in t, M is the basis matrix and G is the geometry vector. Expanding this product using cubic polynomials gives:

$$Q(t) = [x(t) \;\; y(t) \;\; z(t)] = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

The product $T \cdot M$ gives the blending functions for the spline which can be written explicitly as the vector [$\boldsymbol{b_1(t)} \;\; \boldsymbol{b_2(t)} \;\; \boldsymbol{b_3(t)} \;\; \boldsymbol{b_4(t)}$]. Each of the elements G_i is a vector describing the 3D location of a control vertex.

Explicitly multiply $T \cdot M$ and then using the summation formula for matrix multiplication yields the alternative representation for the curve as a weighted sum of basis functions

$$Q(t) = \sum_{i=0}^{4} b_i(t) \cdot G_i$$

where the vector-valued G_i are the weights for the basis functions b_i.

Each segment is joined to the next one at a point, called a *joint*. Figure 3.1 shows a spline with six curve segments defined by nine control vertices represented as squares. The ordering of the vertices along the curve is illustrated by the lines joining them forming a set of control polygons for each curve segment. The joints, shown as circles, separate neighbouring curve segments on the curve.

One important property of splines is local versus global control. A spline with local control has control vertices which, when moved, only affect the section of the curve in the vicinity of the control vertex moved. Control vertices for a spline with global control will affect the entire curve when moved. Another property is interpolating versus approximating of the control vertices. Interpolating splines intersect a subset of its control vertices while approximating splines in general, do not.

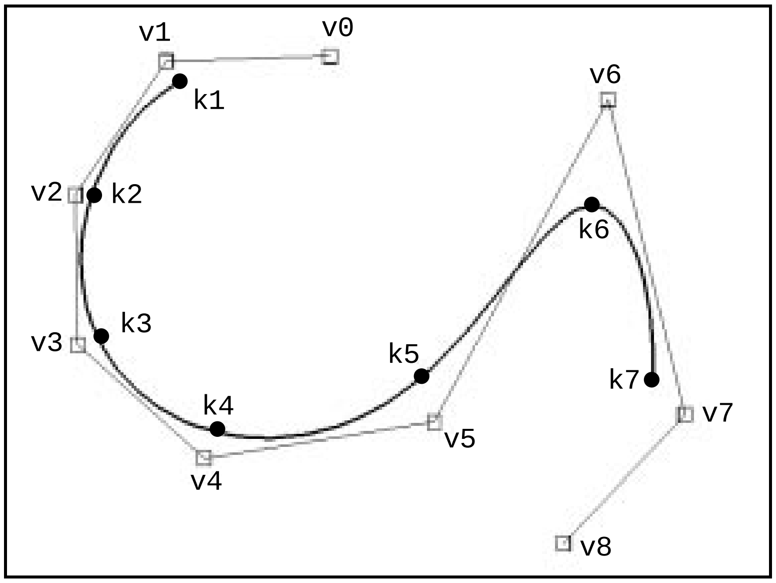


Fig 3.1 A spline with its control vertices (squares) and joints (circles).

3.1 B-Spline Curves

Like many other splines, B-splines approximate curves using piece-wise cubic polynomials. B-splines have local control and are approximating. The minimum cubic B-spline curve is a B-spline curve segment, defined by four control vertices. The segment is constructed by taking a weighted average of the four control vertices. The B-spline blending functions shown in Figure 3.2 illustrate the weighting of the control vertices along a curve segment. These functions are derived from the B-spline basis matrix which can be used to compute the spline curve segment. The basis matrix or blending functions, along with the control vertices, completely define the curve. Figure 3.2 illustrates the use of the blending functions. At *t=0*, the functions give the weights for each control vertex at one end of the curve segment, and at *t=1*, are the weights for the other end of the segment. In the middle of the segment, at *t=0.5*, v1 and v2 get almost half of the weight each, leaving very small weights for v0 and v3. These weights must sum to one. Notice that at the endpoints, the weight for one of the control vertices (v0 and v3, respectively) drops to zero so that the endpoints are a weighted average of only three control vertices.

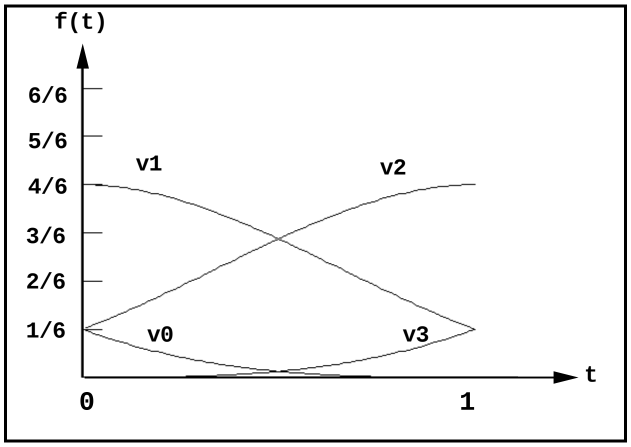


Fig 3.2 B-spline blending functions for a single curve segment as a function of the parameter t.

To approximate a more complicated curve, two or more curve segments may be joined together to produce a B-spline curve providing more degrees of freedom. The second segment uses the nearest three control vertices from the first segment so that a B-spline curve consisting of two segments has five control vertices. Figure 3.1 shows a B-spline curve with nine control vertices and six curve segments. As mentioned earlier, the point on the curve where the two segments are joined is called a *joint*. Because joints occur at the endpoints of curve segments, their locations are calculated from only three control vertices. Because each segment is partially defined by the three vertices from the previous segment, each control vertex may affect up to four curve segments. Figure 3.3 shows the blending functions for four consecutive segments of a B-spline curve. Notice that the control vertex v3 contributes to all four curve segments with a smoothly[3] varying weighting factor. So for a cubic B-spline curve comprising *n* curve segments, *n+3* control vertices are required.

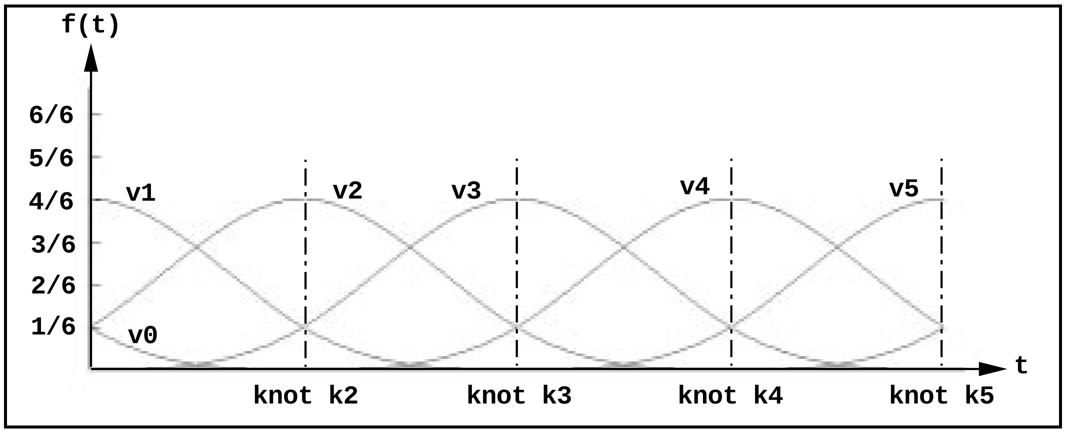


Fig 3.3 B-spline blending functions for first four curve segments.

Each control vertex affects at most four curve segments, verifying its local control property. The fact the curve is created from a weighted *average* of several control vertices shows that it is approximating and not interpolating. A full explanation of B-splines is beyond the scope of this thesis. The interested reader may consult any of the standard texts for additional information [Bart87, Foley90].

[3]This "smoothness" gives the cubic B-spline curve C^2 continuity where two consecutive curve segments meet, i.e., the first and second derivatives at each joint are equal.

3.2 Shape-Matching Paradigm

Spline curves are widely used for interactive curve design in industry. Examples include the design of fonts, the design of motion paths for animation cameras, and automotive and aeronautical CAD. As discussed in previous sections, there are many mathematical formulations for splines. Most of these offer enough freedom to represent a variety of curves, but their behaviours during interactive curve manipulation are diverse. Some of them offer manipulation techniques that are easier to use or are more intuitive, especially for the novice user. To formally compare these different techniques, experiments can be conducted in which subjects are asked to design a number of specified "target" curves using different manipulation techniques. To determine which of the techniques is the best for the interactive design of curves, the amount of time taken to construct the curves and the accuracy obtained can be analyzed.

One major problem with such an experiment is how to describe the target curves to the subjects so that they can attempt to design the curves. In industry, designers are usually not given a comprehensive list of requirements for a curve to be designed, but rather have an idea of the curve in their mind based on some constraints from the application. Although an experiment should be as realistic as possible, subjects must be given a concise description of the target curves to ensure that all subjects have the same goal so that their performance can be judged for correctness. Because of the difficulty in specifying the target curves, we are lead to consider an alternative task. Instead of trying to design curves from some description, subjects are given the target curves and they simply attempt to match them as accurately and as quickly as possible by manipulating a working curve to the target curve with successive manipulation of control points. It is hoped that this curve-matching experiment is equivalent to the curve-design experiment in that the results from the former can be applied to the latter.

3.3 Shape-Matching Experiments

A number of different splines have been tested in previous 2D shape-matching experiments using this paradigm. Some of their properties are reviewed below along with the results of the those experiments. Finally, some background information regarding the splines used in the experiment described in this thesis are given.

The first experiment to use the shape-matching paradigm was conducted by Eric Bosch. He compared the effectiveness of four spline formulations: B-spline, Bézier, Catmull-Rom, and C^2 interpolating. Each of these splines is represented by piecewise cubic polynomials. The Catmull-Rom and C^2 interpolating splines are interpolating while Bézier is both approximating and interpolating. As noted in the previous section, B-splines are approximating. All of the splines offer local control except for C^2 interpolating. The results of the experiment produced no clear favourite, but two results were significant. The B-spline was the most accurate of the formulations and the Catmull-Rom spline was most preferred by subjects. For further details, refer to Bosch's masters thesis [Bosch88] or a subsequent journal article [Bosch93].

Paul Ruest conducted the second curve-matching experiment, using an enhanced version of Eric Bosch's software. He explored the usefulness of the tension parameter available in the rational B-spline and the Beta2 spline. This parameter allows additional control over the curve beyond the actual placement of control vertices. These two splines were compared to B-splines with seven and ten control vertices, but no tension parameter. The rational B-spline and Beta2 each had seven control vertices and a tension parameter for each vertex, offering much more control of the curve than a B-spline with seven control vertices. Results from the experiment revealed that tension was not useful for this task and the overall winner was the B-spline with ten control vertices. Refer to Ruest's masters thesis[Ruest89] or a subsequent journal article [Ruest] for details of the experiment.

From these results, B-splines appear to be a good choice for curve design. Interestingly, the Catmull-Rom spline was most favoured by the subjects and did perform reasonably well in Bosch's experiment. Further insight into these results may be obtained by looking at some of the properties of splines that are relevant to interactive curve manipulation. Both B-spline and Catmull-Rom have local control, often a desirable property. Local control allows the user to work on one section of the curve at a time without disturbing the rest of the curve. A differing property between the two splines is that B-spline is approximating while Catmull-Rom is interpolating. Intuitively, an interpolating spline should be easier to use than an approximating spline because the effect of the control vertices on the interpolating spline is more predictable. The vertices on an approximating spline may potentially lie very far away from the curve, making it difficult to determine which vertex affects a certain section of the curve. This difference does not seem to be very important for 2D curves.

The Catmull-Rom spline seems to have a better set of properties for the curve-matching task over the B-spline, and was preferred by the experiment subjects, but the B-spline was still more accurate. Perhaps the Catmull-Rom spline would have out-performed the B-spline if the target curves were more complicated with self-intersections. Then, the interpolating spline would have a larger advantage over the approximating spline because of the added complexity. This thought leads the way to the current curve-matching experiment, which deals with 3D curves.

Mathematically, 3D splines are not much more complicated than their 2D counterparts. The basis functions remain the same as do the calculations for the spline. There is simply one more coordinate to calculate. However, the style of interaction can be vastly different. Manipulating 3D splines represented as a 2D projection on a computer display is much more difficult than doing so with 2D splines. Even a simple curve may be seen as having a number of intersections in a given view. An interesting question then, is whether a 3D curve-matching task is sufficiently complicated for an interpolating spline to be a major benefit over an approximating spline. The next section describes a method that allows a B-spline curve to imitate an interpolating spline, allowing for a comparison between two different B-spline formulations in Chapter Four.

3.3 Direct Manipulation

"Direct Manipulation" is a term coined by Shneiderman [Shne83] almost a decade ago. He used it to describe computer interfaces that were easy to learn and master, were intuitive for the novice and experienced computer user, and were enjoyable to use. He did not give a precise definition of the term but described it with a variety of examples, a list of properties, and quotes from others describing direct manipulation. Here is a list of properties which he gave:

- Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
- Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.

- Knowledgeable intermittent users can retain operational concepts.
- Error messages are rarely needed.
- Users can immediately see if their actions are furthering their goals, and if not, they can simply change the direction of their activity.
- Users experience less anxiety because the system is comprehensible and because actions are so easily reversible.
- Users gain confidence and mastery because they initiate an action, feel in control, and can predict system responses.

The following quotations were provided to anecdotally describe direct manipulation: “what you see is what you get” and “a representation of reality that can be manipulated”.

The term “direct manipulation” has been used to describe the technique of manipulating a spline curve by picking any point on the curve, dragging it to a new location, and having the curve pass through the new point [Bart89]. In this thesis, direct manipulation refers to a similar technique where a finite number of points on the curve are chosen for manipulation. Before describing the technique, some justification is presented for labelling this technique as “direct manipulation”.

In comparing this technique to the standard manipulation technique for a spline where control vertices (possibly lying far away from the curve) are dragged, direct manipulation appears to have a number of advantages. It should be easier for a novice to learn, more predictable, and “closer to reality”. In curve design, the user has a particular curve in mind that he would like to represent on the computer. As the shape is being formed, the user knows which parts are not quite right and how the curve should deform to fix it. It should be possible to push and pull on parts of the curve itself to achieve this task, not drag points which are off the curve. To understand this, imagine someone working with a stiff wire or a piece of rope on the ground to obtain a certain configuration. The person grabs parts of the wire or rope and bends or moves it to the desired location. Working with a curve represented on the computer should be no different. Given the preceding arguments, it is felt that the spline manipulation technique described in the remainder of this chapter satisfies Shneiderman's definition of "direct manipulation" so the technique will be referred to as "direct manipulation". The experiment presented in Chapter Four quantitatively compares this direct manipulation of a B-spline with standard control vertex manipulation of a B-spline.

The implementation essentially involves creating an alternative set of control vertices providing the user with the illusion that the spline is an interpolating spline as opposed to an approximating spline. These points will be called *direct manipulation points*. When discussing a set of curve manipulation points (control vertex or direct), they will generically be referred to as *control points*. The curve is still defined by the control vertices, so both sets of points must be kept up-to-date during curve manipulation. The implementation can be divided into two parts: the calculation of the locations on the spline of the direct manipulation points and the updating of the control vertices to produce the changes in the curve after modification of the direct manipulation points.

Many choices are available for the selection of the direct manipulation points. Any number of these points may be used and they may be placed anywhere on the curve. In fact, having an "infinite" number of them is not only possible, but very useful. Of course, the number of points is not really infinite because the mouse and the computer display deal with discrete elements. What is meant by "infinite" is that a user may select any part of the curve and modify it by dragging that part to a new position. If this interface is applied to spline surfaces, manipulation would be analogous to sculpting because the user is able to change any part of a surface by simply grabbing a point on the surface and moving it in 3D. This implementation has been used successfully for a hierarchical B-spline surface modeller [Fors90].

The shape-matching experiment presented in this thesis required that there be the same number of direct manipulation points as control vertices in order for the two interaction techniques to be as similar as possible. A variety of locations are possible for these points. The joints of the B-spline curve were chosen. The reason for this will become apparent in the next paragraph. A minor detail is that there are two fewer joints than there are control vertices, making it appear that there is less control of the curve using this implementation of direct manipulation. This was actually not a problem because only a subset of the control vertices were modifiable by the user, so only a similar subset of direct manipulation points was needed. This point is discussed in more detail in the chapter on the shape-matching experiment.

The update of the control vertices is quite simple and, depending on the interaction desired, may be trivial. When a direct manipulation point is moved, at least one and possibly more control vertices have to be updated to reflect the change in the curve. It is the designer's choice which of the control vertices are affected by each direct manipulation point. Of course, none of these points can affect more than four control vertices because points on the spline are determined by at most four control vertices. In the shape-matching experiment's implementation, each direct manipulation point is mapped to only one control vertex. In the blending function for a B-spline curve (Figure 3.3), the weight of any of the control vertices varies from 0 to 2/3. For any point on the spline in which a control vertex gets a weight of more than 1/2, that control vertex has the dominant influence on that point of the spline. We chose to use the joints of the spline for the locations of the direct manipulation points. It is at these points that a control vertex assumes its maximum weight of 2/3, so this is the lone control vertex that is updated when the corresponding direct manipulation point (or joint) is moved. This is the technique used by Forsey. The direct manipulation points are referred to as “edit points” [Fors90]. Because only one control vertex is updated, the same local control is retained even when using direct manipulation. If, on the other hand, we chose to map our direct manipulation points (the joints) to all three control vertices which affect it, spline manipulation would be less localized. Specifically, six curve segments surrounding the joint would be modified instead of just four.

If every point on the spline is a direct manipulation point, there are two choices for the update of the control vertices. One can update every control vertex which affects the point picked on the spline or one can update the single control vertex which has maximum influence over the point picked. In the first option, the amount to update each of the four control vertices can be determined from the weights for that point. If only one control vertex is updated, a problem arises in the middle of a curve segment where two control vertices are “tied” for the maximum influence (i.e., there is no dominant control vertex). One can either chose one vertex by default or use both of them. This issue does not arise for the joints.

In our implementation, looking at each direct manipulation point ($\boldsymbol{D}_i$) and corresponding control vertex ($\boldsymbol{C}_i$) pair, we can derive an equation relating them:

$$\frac{1}{6}\boldsymbol{C}_{i-1}+\frac{4}{6}\boldsymbol{C}_i+\frac{1}{6}\boldsymbol{C}_{i+1}=\boldsymbol{D}_i$$

The control vertices $\boldsymbol{C}_{i-1}$, $\boldsymbol{C}_i$, and $\boldsymbol{C}_{i+1}$ determine the value of $\boldsymbol{D}_i$ which is also joint k_i. The weights or coefficients used for the control vertices are derived from the B-spline blending functions. If $\boldsymbol{D}_i$ is moved by Δ_i, we get the following equation:

$$\frac{1}{6}C_{i-1}+\frac{2}{3}(C_i+\frac{3}{2}\Delta_i)+\frac{1}{6}C_{i+1}=D_i+\Delta$$

So, according to the second equation, if $\boldsymbol{D}_i$ is moved by Δ_i, the spline should be updated by moving $\boldsymbol{C}_i$ by $1.5\Delta_i$. The implementation is simple. When any of the direct manipulation points is moved, the corresponding control vertex is moved by one and a half times that amount.

Notice, however, that because the position of a direct manipulation point is defined by three control vertices, when one direct manipulation point is moved causing a control vertex to be moved, the neighbouring direct manipulation points must also move in response to that control vertex's new value. This behaviour is the one major drawback of our direct manipulation technique. We can hope that the results of the shape-matching experiment will reveal whether this is significant for interactive curve manipulation. This is the focus of the next chapter.

Chapter 4

Direct Manipulation Experiment

A curve-matching experiment comparing the effectiveness of direct manipulation of 3D B-spline curves with that of control vertex manipulation was conducted to test the hypothesis that although direct manipulation may not provide a significant advantage when matching 2D curves, it would prove to be superior for 3D curves. Control vertex manipulation is indirect because these points may potentially lie very far away from the spline curve itself. When working with 3D curves represented on a 2D display, this indirection could be a major barrier to overcome in successfully manipulating the curves.

In the experiment, the 3D task that subjects performed was the matching of 3D spline curves. Like most 3D tasks, this one may be divided into the sub-tasks discussed in Chapter 2. The first sub-task, the manipulation of points, corresponds to the selecting and moving of control points, either control vertices or direct manipulation points. The second sub-task, modification of object orientation, corresponds to the 3D rotation of the curves on the display. The third sub-task, rendering of the 3D scene, corresponds to the choice of depth cues for the 3D curves.

4.1 Environment

The experiments took place in the GraFiC Laboratory in a corner separated from the rest of the lab with black curtains. The enclosed area, approximately three meters by three meters, contained an office chair, a table, and a Silicon Graphics Iris 4D workstation, model 240VGX. The computer was actually located in a machine room, so only the monitor, keyboard, and mouse were present. Because the mouse was used for the vast majority of the experiment, it was placed directly in front of the seated subject, near the edge of the table. The location of the mouse and its optical mouse pad could not be changed by the subject because the pad was secured to the table top with tape. This was done to prevent it from sliding during the experiment. The keyboard was placed between the mouse and the monitor. A single lamp with a 60 watt household light bulb was placed behind the monitor and pointed upwards and away from the subject. The laboratory itself was dimly lit, so with the curtains closed, screen glare was negligible. Contrast on the monitor was set to its maximum as was the intensity. The terminator switches on the back of the monitor were all set to 75 ohm.

4.2 Target and Controlled Curves

Both the target and the controlled curves were defined by B-splines with nine control vertices. The position of each control vertex for the target curve was randomly generated. There were twenty target curves used in the experiment. After the set of target curves were created, they were screened for their level of difficulty, but none were rejected. The four practice target curves were not randomly generated, but were carefully chosen so that the initial practice trial was straightforward, with each successive trial gradually becoming more difficult.

Several options were available for the initial position of the controlled curve. In previous 2D curve-matching experiments [Bosch87, Ruest89], the controlled curve was laid out diagonally across the screen. One difficulty in those experiments was the matching of the endpoints of the target curves. For many of the matches, the control points were not properly spaced along the target curve, so although the match was acceptable, the controlled curve would protrude past the end of the target curve. Some subjects resolved this problem by wrapping the controlled curve back onto the target curve. Matching the endpoints in a 2D curve-matching task is already difficult; for 3D curves, it is nearly impossible for a novice user. To simplify the matching of the curves for this experiment, it was decided that the endpoints of each controlled curve would be permanently attached to the target curve so that only the middle section needed to be manipulated. To achieve this, the three control vertices at either end of the controlled curve were assigned to the corresponding control vertices in the target curve, and thus were not modifiable by the subject during the trial. With this constraint in place, each trial involved matching the three middle control points of the controlled curve to those of the target curve, leaving the other six control points unchanged. Only the three manipulated control points of the controlled curve were displayed. The coordinates for the control vertices for each of the target curves are listed in Appendix E. Although the rendering of the curves will be different depending on the software and hardware used, this data does allow an exact reproduction of the 3D shape of the curves for similar experiments that may be run in the future.

4.3 Experiment Procedure

The experiment was divided into two independent sessions for each subject. One session dealt with B-spline curves using standard control vertex manipulation while the other dealt with B-spline curves using direct manipulation. The order in which the two sessions were performed was random, with half of the subjects doing the sessions in each order. Subjects were required to complete an entire session in one sitting. They were not allowed to complete both sessions in one day, but the second session had to be performed during the same work week as the first. These constraints were enforced to allow short-term learning to occur between the two sessions and to avoid long-term learning.

Prior to the start of the first session, each subject filled out a personal information form and a consent form. Each session comprised an on-line tutorial (which was optional for the second session), twenty-four trials that were separated into four initial practice trials and twenty recorded trials, and a short subjective rating session. At the end of each session, subjects were given a comments form to fill out. Copies of these forms can be found in Appendix B.

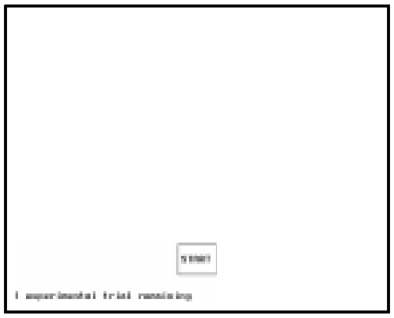
During each trial, the subject was presented with two 3D B-spline curves, one blue and one red, drawn inside a shaded 3D bounding box filling the entire screen. The blue curve remained fixed throughout the trial and was called the *target* curve. The shape of the red curve could be changed, and was therefore called the *controlled* curve. This curve could be manipulated by moving any of the three control points, represented as small black cubes. If the trial was part of the direct manipulation session of the experiment, the control points would all lie on the controlled curve. If it was a control vertex session, they would not. The task during each trial was to modify the controlled curve so that it matched the target curve as closely as possible in a short amount of time. Because the curves were three dimensional, it was necessary to view and manipulate the curves in different orientations to complete the match.

To move a control point, the subject would place the cursor over the desired point, press and hold down the left mouse button, and drag the cursor, moving the control point along with it. To change the orientation of the curves, the subject was given the ability to rotate the curves by ninety degrees in any direction using the middle mouse button. Depending on which region of the screen the cursor was in when the button was released, the curves would rotate in one of the four directions. The rotation was not instantaneous, but took about a second to complete. This helped the subject visualize the shape of the curves. The right mouse button was used to end the trial. Details of the user interface can be found in the tutorial reproduced in Appendix A.

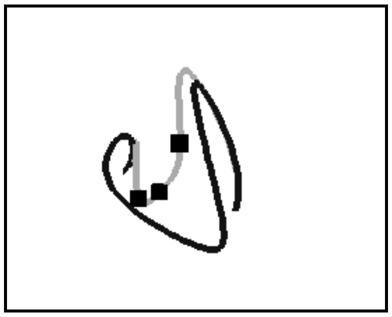
During the course of a trial, the subject was required to do the following: select a START button on the screen which starts the timer and displays the curves for matching; match the curves; stop the timer by picking the appropriate item from a pop-up menu; and, finally, give a subjective rating of the match. The subject was allowed to pause as long as desired between trials. Figure 4.1 illustrates the execution of a trial.

To help analyze the subjective ratings for each trial, subjects were given a rating session after the completion of all of the trials. During this session, they were simply required to study a set of matched 3D curves and to rate them in the same manner as in the trials. There were twenty pairs of curves in a set, the first group (a group consisted of four pairs of curves) were common to all subjects, while the next group consisted of the 4th, 8th, 12th, and 16th matches performed by the subject displayed in the same orientation in which the subject last viewed them during the trial. The third group of curves were the same as the second group except that the view had been rotated by 90 degrees about the z axis. The fourth and fifth group used the same pairs of curves again, but started with a random view.

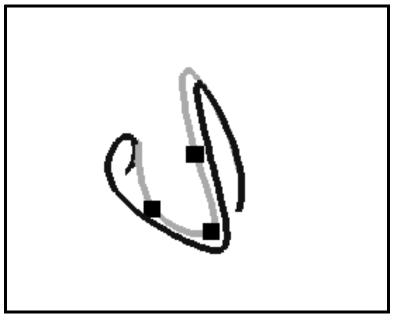
The software used to conduct the experiment is described in Chapter Five. It is available to other researchers for use in similar hardware environments.



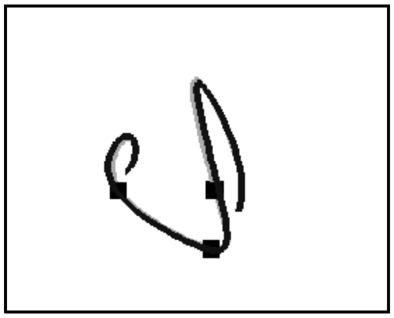
a) START button.



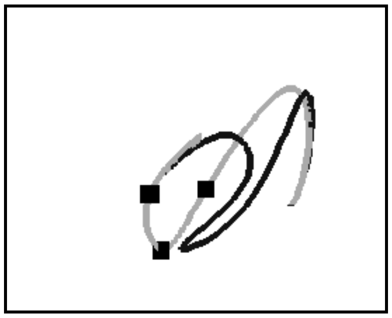
b) Initial view of curves.



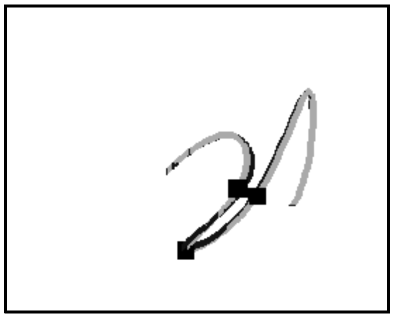
c) After moving the middle control point.



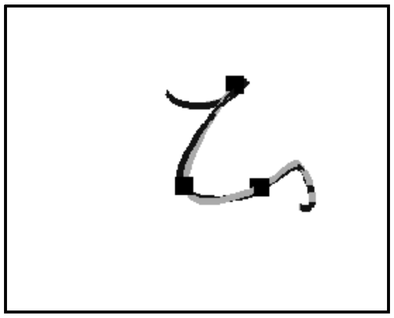
d) After moving other control points.



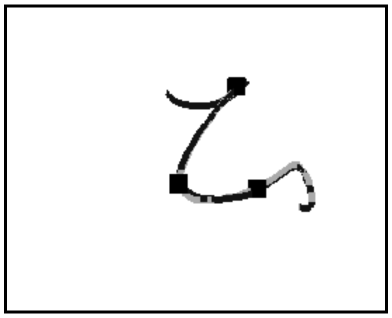
e) After rotating the view to the right about the vertical axis.



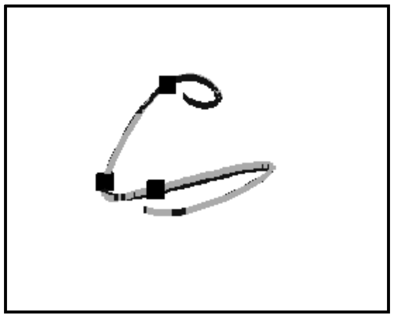
f) After moving more points.



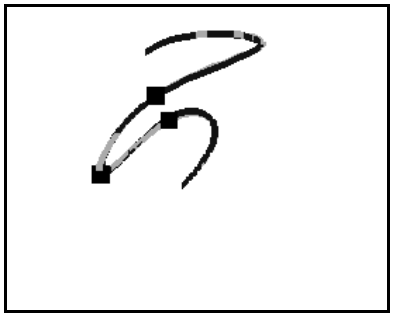
g) After rotating the view up about the horizontal axis.



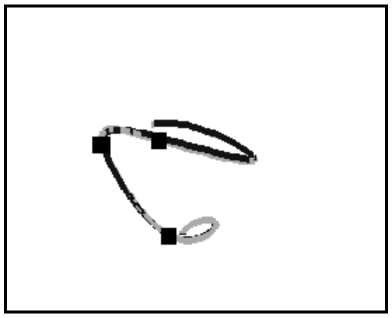
h) After moving the middle point.



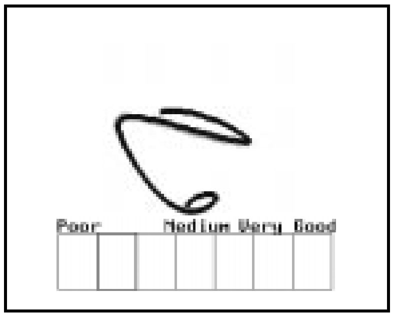
i) After rotating the view to the right about the vertical axis.



j) After rotating the view up.



k) After rotating the view up.



l) Stop, rate match.

Fig 4.1 A synopsis of a complete curve-matching trial.

4.4 Subjects

Sixteen subjects took part in the experiment. Twelve were male and four were female. Five of the subjects had completed engineering undergraduate degrees, six were engineering students, three had science undergraduate degrees, one was a chemical technician and one was a school teacher. Their ages ranged from 21 to 32. Every subject had some kind of computer experience, and all but one had some mouse experience. Seven subjects had used a curve drawing program previously and four had some knowledge of spline mathematics. The majority of the subjects were enlisted with advertisements placed in the Electrical Engineering building and the Pulp and Paper Research Centre. A detailed description of the subjects is provided in Appendix C.

Originally, subjects were obtained from a sign-up sheet attached to an advertisement posted in the Psychology building. On the sign-up sheet, potential subjects left their names and phone numbers. However, some of these people were very difficult to reach, and of those who scheduled times to perform the experiment, many either cancelled or did not show up. Only two people actually arrived for the experiment, both of whom did not follow the experiment instructions correctly and thus could not continue with the experiment. From this unsuccessful experience, it was learned that interested subjects should contact the experiment supervisor, not vice versa. It was also concluded that for this experiment, pre-requisites should include some kind of technical background and a good comprehension of written English.

After interviewing some pre-experiment subjects and the two unsuccessful subjects from the Psychology sign-up sheet, it became obvious that a random pool of subjects would not work. The problem was not that the spline manipulation was too difficult, but that the interpretation of the curves as 3D objects being represented on the display was very daunting for some of the people. In fact, this was the problem that the two unsuccessful subjects had. They failed to understand that the curves were three dimensional, and that matching the curves required matching in every dimension. They only matched the curves in one view and stopped. The background needed for this experiment was not necessarily a science or an engineering degree, but some experience in visualizing 3D objects. Such experience could be obtained in a drafting course, a molecular chemistry course, a vector calculus course, or perhaps even a metal or wood working course. It was hoped that Engineering students (or alumni) would have this background.

4.5 Analysis of Experimental Curve-Matching Data

For this curve-matching experiment, the following three hypotheses were investigated:

H1. Direct manipulation is “better” than control vertex manipulation for 3D curve-matching.

H2. Learning has a major effect on the performance of this task.

H3. Direct manipulation can be learned faster than control vertex manipulation.

Results from the experiment were collected from statistical analysis of the time and error data, subject comments forms, playback data, and verbal discussions with the subjects. Other results could be obtained at a later time because every trial was recorded to a playback file, and thus could be played back in real time to tabulate more data. Hypothesis 1 was verified. Direct manipulation was faster than control vertex manipulation. Hypothesis 2 was also verified, and was in fact the strongest effect in the curve-matching experiment. There was no statistical evidence for Hypothesis 3.

The majority of the analysis looked at the time and error results from the experiment. Two types of analysis were used, the t-test and the analysis of variance (ANOVA). Literature on these tools can be found in any introductory statistics text [Devo82]. The t-test was used to compare the means from two samples of data and determine if there was enough evidence to conclude with some minimum probability that the two means should be different. For this analysis, that probability was 0.95. In other words, the probability (called α) of concluding a difference in the means when there really was not one, was 0.05 or less. When a conclusion can be made that the means are different, that result is said to be *significant*. At the same time, a 95% confidence interval can calculated for the difference in the means. An ANOVA was used to compare the interaction of two factors in a set of samples. Again, an α value of 0.05 or lower was used to decide if a result was significant. Both types of analysis assume that the samples are normal and homogeneous (each block has the same variance). However, in [Glass72], it is shown that the results of a t-test or ANOVA are still valid when the data is heterogeneous, as long as each block had the same number of data values. For completeness, the sample variances for each block were calculated and checked for extreme deviations.

The experiment results can be broken up into four blocks upon which the analysis was based. These included control vertex sessions performed as first sessions, control vertex sessions performed as second sessions, direct manipulation sessions performed as first sessions, and direct manipulation sessions performed as second sessions. These blocks are later referred to as CV1, CV2, DM1, and DM2 respectively. Each of these blocks consisted of twenty trial results from eight subjects. For a given

Table 4.1 Time and error results.

Results	CV1	CV2	DM1	DM2
Median Time (sec.)	161	125	140	110
Median Error (pixels)	13.0	8.9	9.5	10.3

subject, the twenty trials involved matching a fixed set of twenty target curves given in a random permutation. For each analysis, a block of results was averaged by subject, trial, or item. For example, in a *subject analysis*, the average for each subject over all trials was calculated and that data was used in the analysis. Because there were eight subjects in a block, the subject analysis dealt with eight values. For a *trial analysis*, averages over all eight subjects for each of the twenty trials were used. Trials are labelled by their chronological order for each subject. For an *item analysis*, averages over all eight subjects for trials containing each target curve were used. There were twenty different target curves so twenty values for each block were used in this analysis. In most cases, grouping the data in each of these ways produced similar results unless there were outliers in the data. Thus, comparing these results helped reveal possible outliers.

There are a number of techniques for averaging a set of data. The mean is the most common measure. However, it is not very useful when the data contains outliers, or extreme data points. The mean should especially be avoided when the data is bounded on one side, as is the case with time data, because outliers are certain to bias the mean in only one direction. Two other averaging measures, the median and the mean of the log transform of the data, are less affected by outliers. For this experiment, the median was chosen over the mean of the log transform because transforming data is more complicated when reporting results of means and confidence intervals and the median analysis did provide a satisfactory analysis.

Table 4.2 summarizes the data. Note that these averages are different from those given in the next sections because they are medians computed over the entire block of data, while those in the next sections are means of medians across trials or across subjects for the block.

4.5.1 Testing Hypothesis 1 − Analysis of Spline Manipulation Technique

To determine which technique performed best in this experiment, the t-test was used to analyze the time and error data. Because of the possible learning factor involved in the experiment, the analysis compared CV1 and DM1 and ignored CV2 and DM2. This data is similar to that which would be generated in a between subjects analysis because the eight subjects who produced data for CV1 were different from the eight subjects in DM1. A trial analysis of time was significant with $\alpha = 0.01$. The mean values for CV1 and DM1 were 161 sec. and 140 sec., respectively, with a difference of 21 sec. A 95% confidence interval for the difference in mean times was (6.6 sec., 35 sec.). A subject analysis was also performed, but because of large variances between subject scores, not enough data was available to produce a significant result. A trial analysis of the error generated a significant result with $\alpha = 0.01$ and mean errors of 13.0 and 9.5 pixels for CV1 and DM1. So, direct manipulation was superior for subjects with no prior experience in performing this task.

An alternative question is: which technique is better for experienced subjects? For this analysis, "experienced" is defined as having previously matched curves with either technique, so a comparison between CV2 and DM2 was made. A trial analysis of time was significant with $\alpha = 0.01$ with respective mean times of 125 sec. and 110 sec. Analysis of the error did not produce significant results. Again, direct manipulation proved to be the better technique.

4.5.2 Testing Hypothesis 2 − Analysis of Learning Effects

To study the effects of learning, t-tests were performed on CV1 vs. CV2, DM1 vs. DM2, the first ten trials of CV1 vs. the last ten trials of CV1, and similarly for CV2, DM1, and DM2. Also, graphs of the data with regression analyses were produced. For CV1 vs. CV2, the trial analysis of time was significant with $\alpha = 0.001$ with mean times of 161 sec. and 125 sec., a difference of 36 sec. A 95% confidence interval for the difference is (23 sec., 47 sec.). A subject analysis of time was significant with $\alpha = 0.02$. A trial analysis of error was significant with $\alpha = 0.002$, and mean errors of 13.0 and 8.9 pixels. For DM1 vs. DM2, a trial analysis of time was significant with $\alpha = 0.001$ with mean times of 140 sec. and 110 sec. A 95% confidence interval for the difference is (18 sec., 42 sec.). A trial analysis of error showed no significant results.

Similar analysis was performed on the first half of a block vs. the second half of the block. None of the blocks produced significant results except for CV1. For that analysis, significant results were found for both the trial analysis of time and of error for $\alpha = 0.05$. The mean times for the first and second half were 172 sec. and 151 sec., while the mean errors were 15.6 pixels and 11.5 pixels.

These results show that the learning factor is present across sessions and may in fact be stronger than the technique factor. The differences in mean times for CV1 vs. CV2 and DM1 vs. DM2 were almost double those in the technique analysis. Within the blocks, significant effects in learning were only found in CV1.

To further analyze the learning factor, a number of graphs were plotted and regression lines were fitted to the point data. Figures 4.2 and 4.3 are point plots of the raw time data for CV1 and CV2 (Figure 4.2) and DM1 and DM2 (Figure 4.3). Figures 4.4 and 4.5 contain plots for mean trial times for each of the four blocks. Figures 4.6 and 4.7 show the median trial times. These six graphs are repeated for the error data in Figures 4.8 to 4.13. The twelve graphs plot the trial number vs. trial time (in seconds) or trial error (in pixels). The trial numbers take into account the four initial practice trials, and the trials from a subject's second session are numbered so that they continue from the first session and again take into account the four initial practice trials. Thus the first trial of the first session is trial five and the first trial of the second session is trial twenty nine. All of the graphs contain linear regression lines as well. The regression lines for the raw data (Figures 4.2, 4.3, 4.8 and 4.9) are the same as the corresponding regression lines for the mean trial data (Figures 4.4, 4.5, 4.10 and 4.11). The data points in the graphs in Figures 4.6, 4.7, 4.12 and 4.13, the median trial data, are the data values that were used in the trial analysis for the t-tests. Figure 4.14 includes the regression lines for the mean trial times for each of the four blocks while Figure 4.15 includes the regression lines for the errors.

The equation for a regression line provides information about the amount of learning and the level of difficulty of the task. A large y-intercept indicates a difficult task requiring more time to complete or a higher level of error. A large negative slope represents rapid learning or improvement. To analyze the significance of these lines, hypothesis tests were performed to determine if the null hypothesis of slope = 0 could be rejected. For the median time data (Figures 4.6, 4.7, and 4.14), all of the regression lines had a negative slope but none produced a significant result. For the median error data (Figures 4.12, 4.13, and 4.15), a significant result was obtained only from CV1 with $\alpha = 0.05$ and a 95% confidence interval for the slope of (-0.47, -0.05).

4.5.3 Testing Hypothesis 3 − Interaction between Learning and Technique

A two factor ANOVA was performed to study the interaction between the technique factor and the learning factor. For the trial analysis of time, both the technique and the learning factor were significant for $\alpha = 0.01$, with the result for the learning factor being much stronger. For the subject analysis of time, only the learning factor was significant at that level of α. In both analyses, there was no evidence of interaction. So, for the time data, the two factors were independent. Learning affected the two techniques in similar ways and vice versa. In contrast, for the trial and subject analysis of error, the technique and learning were not significant, but the interaction factor was. Learning affected the error of one technique, control vertex manipulation, more than it did direct manipulation. In fact, in comparing the plot in Figure 4.13, the median error for each trial did not change appreciably from DM1 to DM2.

4.5.4 Comments from Subjects

The comment forms collected information about match strategies, rating criteria, curve manipulation technique rankings, and general comments. The comments on the match strategies can be divided into three groups. They were either strategies performed prior to actual curve matching, general matching techniques, or some specific ordering to the manipulation of the control points. The first group of strategies included rotating the initial curves and looking for the easiest view to begin matching, spreading out the control points towards the perimeter of the match area and away from the target curve, and rotating the curves several times to visualize the 3D shape of the curves. About one quarter of the subjects mentioned that they looked for the easiest view before matching, and from reviewing the playback of some trials, at least half of the subjects did use that strategy some of the time. Some subjects matched the curves by manipulating the control points starting from one end of the curve to the other, while others started at both ends and worked their way towards the middle. General matching techniques included rough matches in the first few views and then fine tuning by rotating through several more views, or obtaining accurate matches for the first couple of views and finishing the match with only a few more rotations. For the latter method, some subjects discovered that after a match in the first view and one ninety degree rotation, matching in this view required only horizontal or vertical movement of the control points, but not both. At this point, subjects had a good match, and with only one or two more views would finish the trial.

The subjective ratings were not well received by the subjects. Over half of them complained that they did not have any kind of a scale on which to base their initial matches. Throughout the experiment, they had trouble using the entire rating scale as instructed in the tutorial. Subjects usually matched until a certain accuracy was reached so that all of their matches received similar ratings. About one quarter of the subjects used the trial time and the difficulty of the match as part of the criteria in rating the match, enabling them to rate on a slightly wider scale.

Subjects were given three questions in which they were asked for a preference between the two curve manipulation techniques. The questions and the results are summarized below:

Q1. In which session did you find the control of the curves easiest?

Q2. In which session did you think that you were the most successful in your final matches?

Q3. Which session did you enjoy the most?

Table 4.2 Comment form results

Question	CV 1st Session	DM 1st Session	Total Score
Q1: easiest	6 DM vs. 2 CV	6 DM vs. 1 CV	12 DM vs. 3 CV
Q2: most successful	4 DM vs. 2 CV	4 DM vs. 3 CV.	8 DM vs. 5 CV
Q3: most enjoyable	5 DM. vs. 2 CV	6 DM vs. 0 CV	11 DM vs. 2 CV

The results were divided into two groups of subjects, those who used control vertex manipulation in the first session and those who used direct manipulation. Some of the responses to these questions indicated a tie or no preference between the two techniques. These responses were simply omitted from the table. For the first and third question, direct manipulation was the overwhelming favourite. The score for the second question was much closer because, as discussed previously, most subjects were equally successful with all of their matches. The difficulty of the trial, whether due to the technique or not, was reflected in the match time not the match quality. Some of the subjects commented that their choices were affected by the learning factor. They felt that their second session was more successful, regardless of which technique was used, because they were more familiar with the task. The results for these subjects confirmed that their second sessions were, in fact, more successful.

Finally, the miscellaneous comments provided by the subjects echoed some of the points already discussed. Direct manipulation was more automatic while control vertex manipulation was harder in that without prior experience, one did not know which point affected a particular region of the curve. The rating of the matches was very difficult in the early trials of the experiment, but became easier in the second session. The learning factor significantly affected the match strategy. Two subjects claimed that direct manipulation was easier for approximate matches while control vertex manipulation was superior for fine-tuning. The latter observation may be due to the fact that when a direct manipulation point is moved, its neighbouring points move as well, making fine adjustments more difficult.

4.6 Discussion of Results

The match times for each trial produced the most interesting results as evidenced in the analysis in the previous section. The match errors provided some insight into the match strategies of the subjects. The error metric used was the sum of the Euclidean distances between the location of the target curve's control vertices and the controlled curve's control vertices. The units used for the distances were the world coordinates of the vertices that were bounded by a unit cube. These values were then converted to screen pixels by scaling them by 500, the approximate length of a side of the unit cube. This metric was initially implemented because of its simplicity. However, there was some concern as to its accuracy as an error metric for this task because the subject does not base the match on the control vertices, but rather the relationship between the two curves.

In general, trial performance was a trade-off between match time and accuracy. In most cases, the trials were not extremely demanding, and a fairly constant, high level of accuracy was maintained. Any effects due to learning or difficulty of the target curve were observed in the trial time alone. For some subjects, however, having little curve-matching experience and faced with the less intuitive control vertex manipulation, early trials in CV1 did prove to be extremely demanding. In these cases subjects were forced to compromise the desired level of accuracy for a reasonable match time.

Preliminary work with test subjects revealed that for each subject, upon mastering the manipulation techniques, the match errors did not vary considerably. Each subject had his own idea of an adequate match which may have partially depended on his amount of patience or stubbornness. For each trial, the curves were simply matched until that level of accuracy was obtained. So subjects essentially kept the error constant for each trial and varied the match time. For this reason, the initial error metric was used instead of investigating other possibilities. Of course, a new metric could have been implemented even after the experiment by extracting the final values for the control vertices of the controlled and target curves. The subjective ratings did not prove to be very useful. Subjects found it very difficult to rate the early trials because they did not know what a good or poor match looked like. The written comments offered important information about match strategy. Finally, the playback data verified the subjects' written comments about strategy as well as revealing other trends and strategies during matching.

Two minor problems were encountered during the experiment. The first problem was the misunderstanding of the initial subjects about having to perform a 3D match as opposed to a 2D match. Updating the tutorial and screening the subjects as discussed in Section 4.4 appeared to be an adequate solution. The second problem was the fact that the computer which was used for the experiment was connected to the network in the Department of Computer Science, allowing other users to log in and use it. During each experiment, the list of processes running on the experiment computer was monitored periodically. On two separate occasions users logged in to the computer and started compute-intensive processes, momentarily slowing down the experiment software.

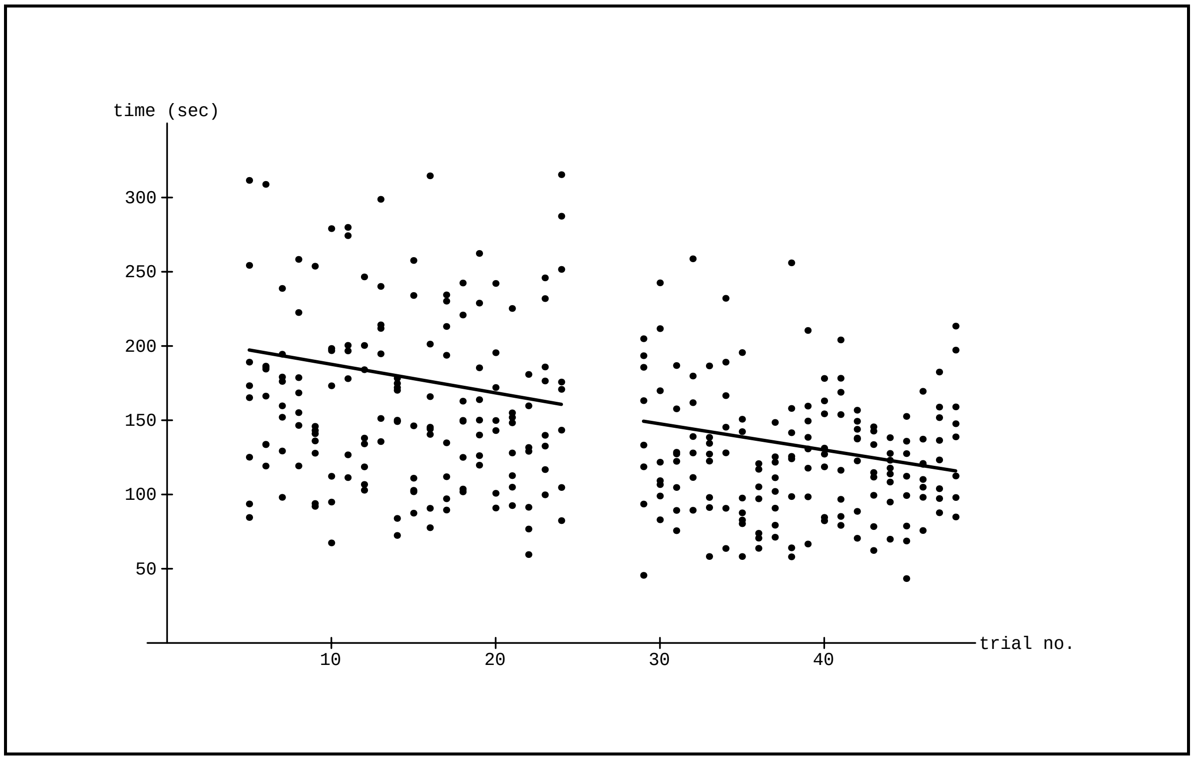


Fig 4.2 Raw time data for control vertex manipulation (CV1: trials 5-24 and CV2: trials 29-48).

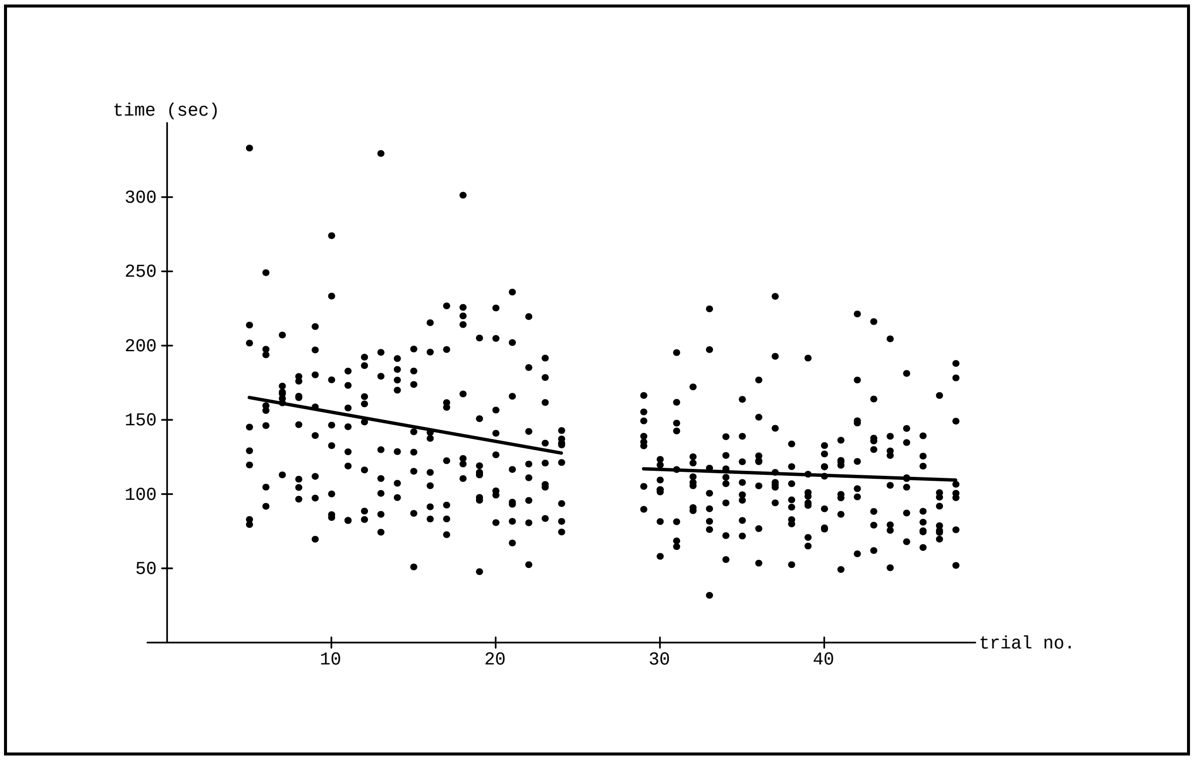


Fig 4.3 Raw time data for direct manipulation (DM1: trials 5-24 and DM2: trials 29-48).

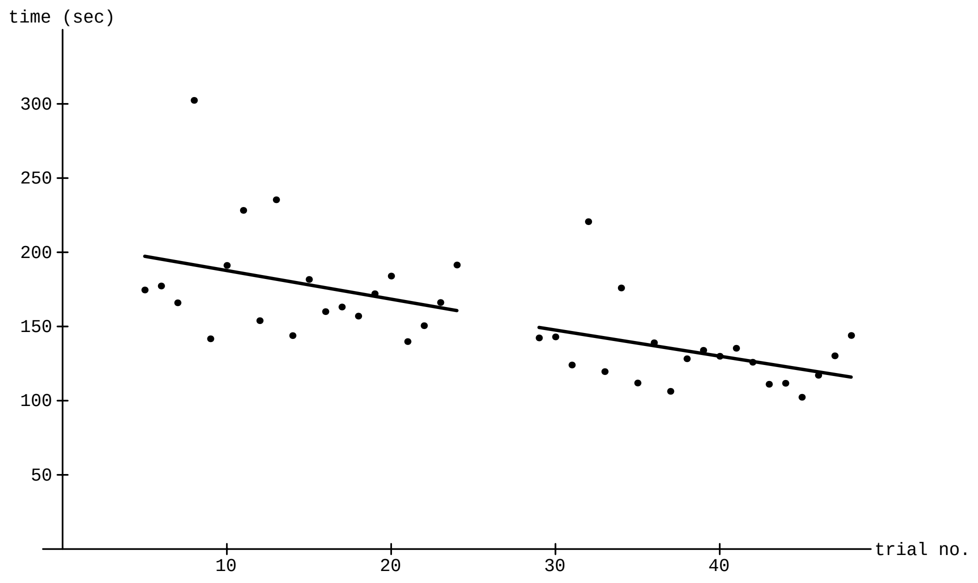


Fig 4.4 Mean trial times for control vertex manipulation (CV1: trials 5-24 and CV2: trials 29-48).

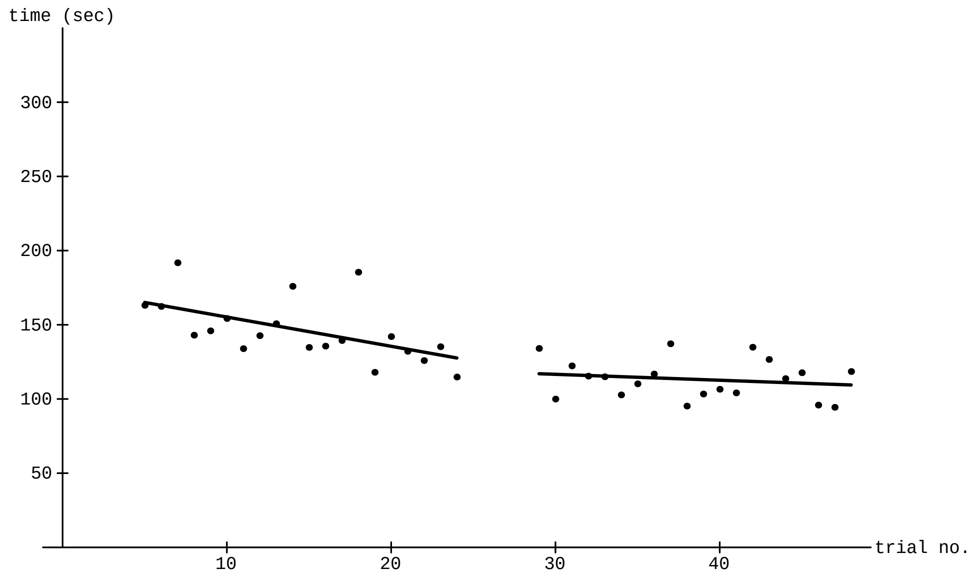


Fig 4.5 Mean trial times for direct manipulation (DM1: trial 5-24 and DM2: trial 29-30).

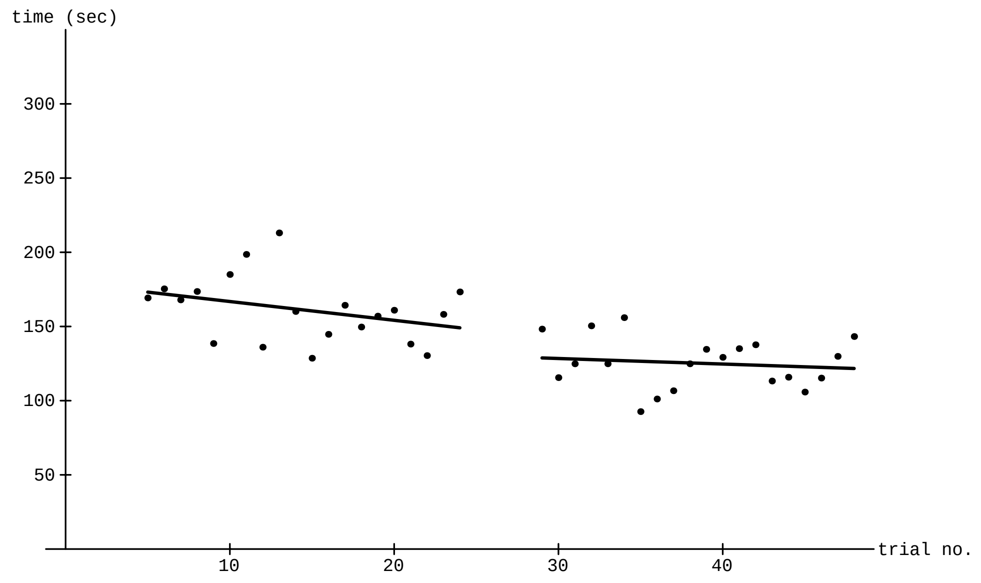


Fig 4.6 Median trial times for control vertex manipulation (CV1: trials 5-24 and CV2: trials 29-48).

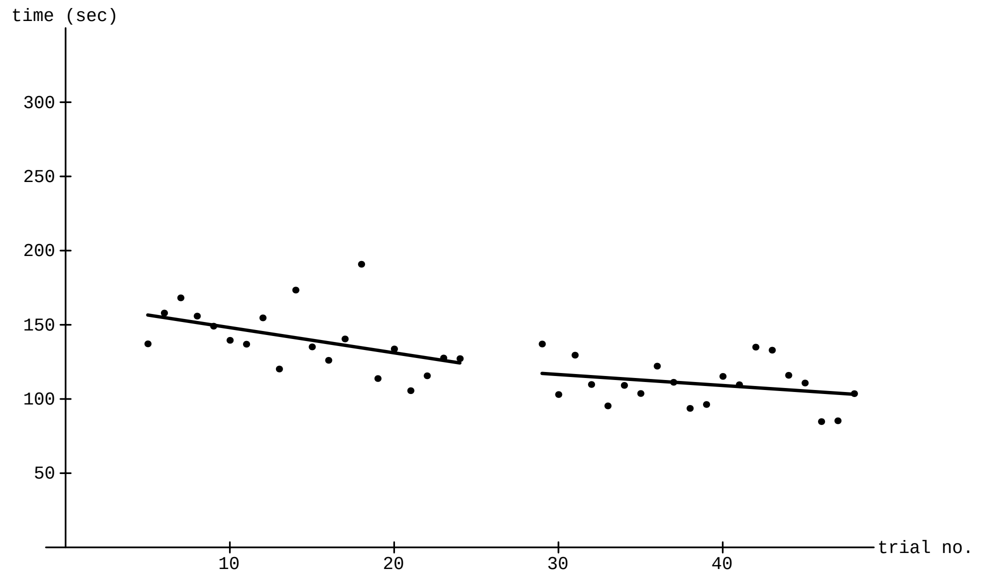


Fig 4.7 Median trial times for direct manipulation (DM1: trials 5-24 and DM2: trials 29-48).

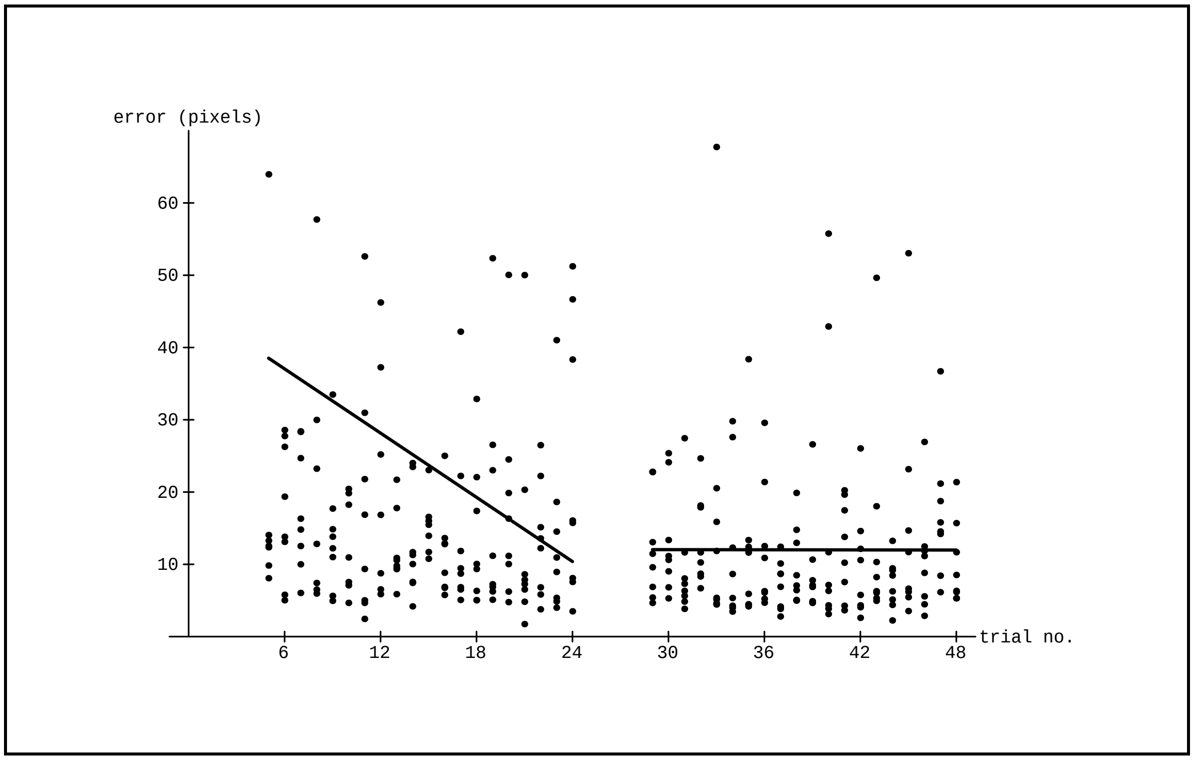


Fig 4.8 Raw error data for control vertex manipulation (CV1: trials 5-24 and CV2: trials 29-48).

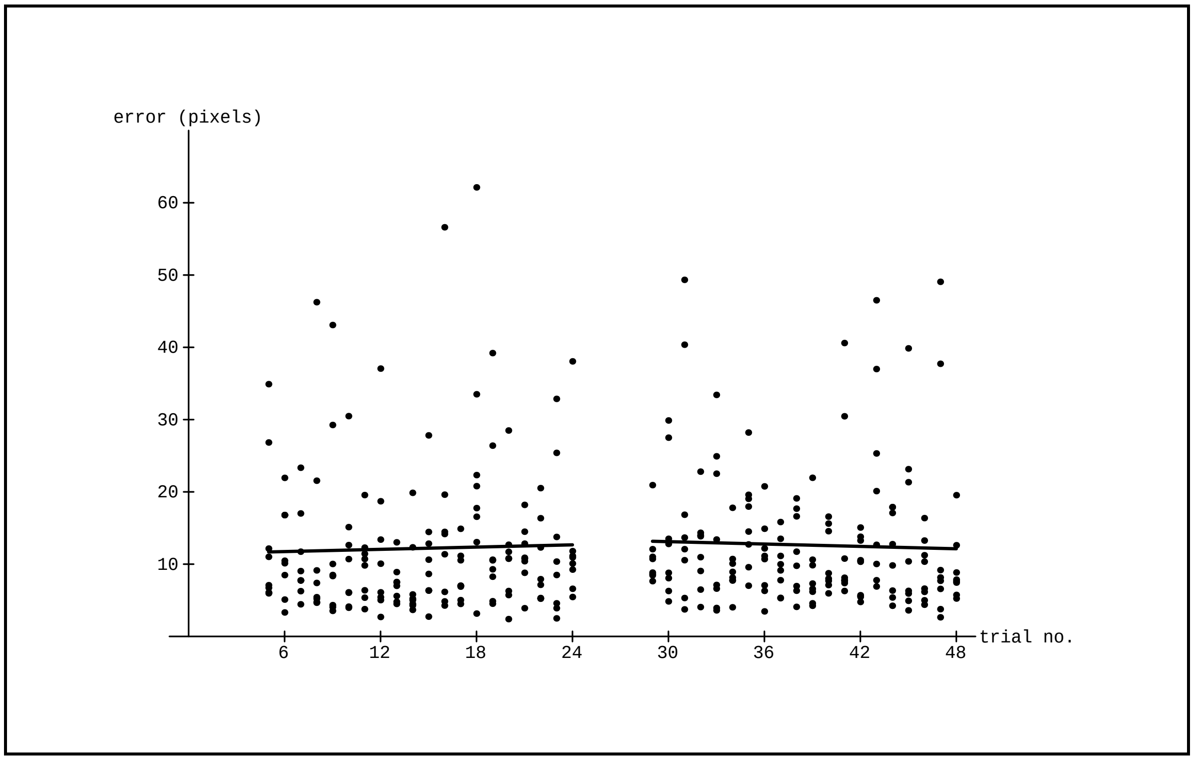


Fig 4.9 Raw error data for direct manipulation (DM1: trials 5-24 and DM2: trials 29-48).

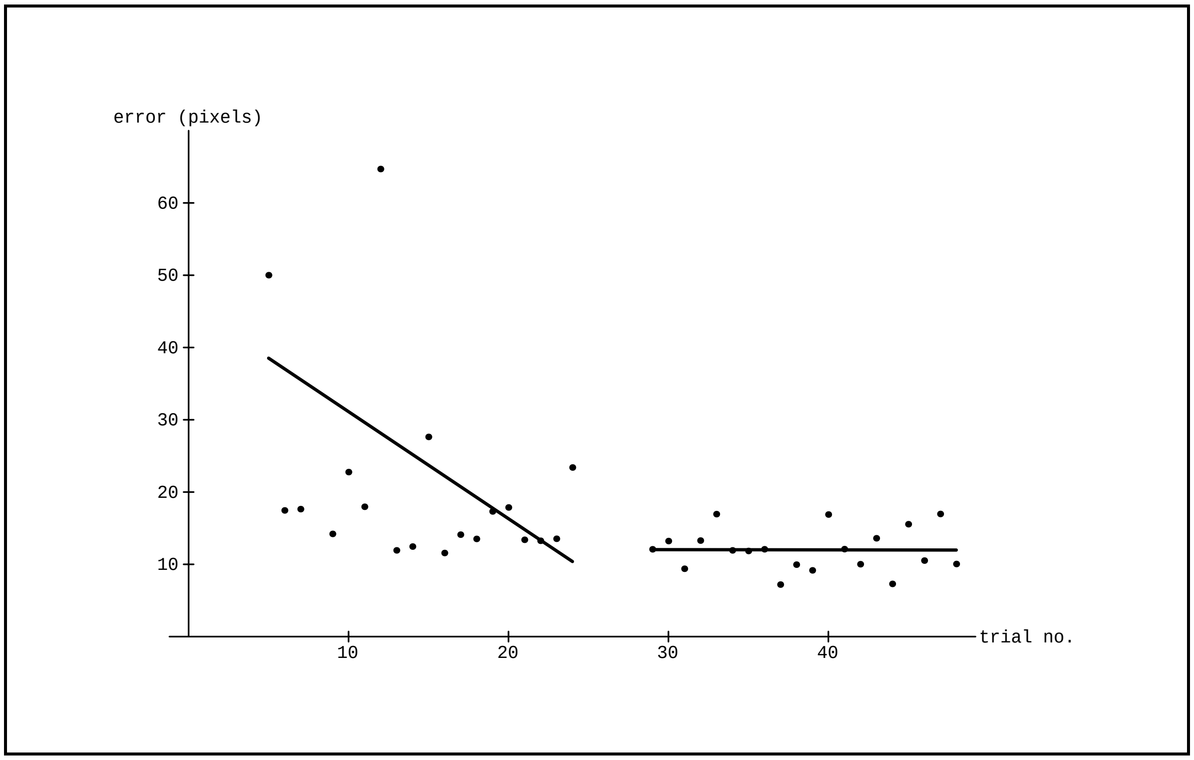


Fig 4.10 Mean trial errors for control vertex manipulation (CV1: trials 5-24 and CV2: trials 29-48).

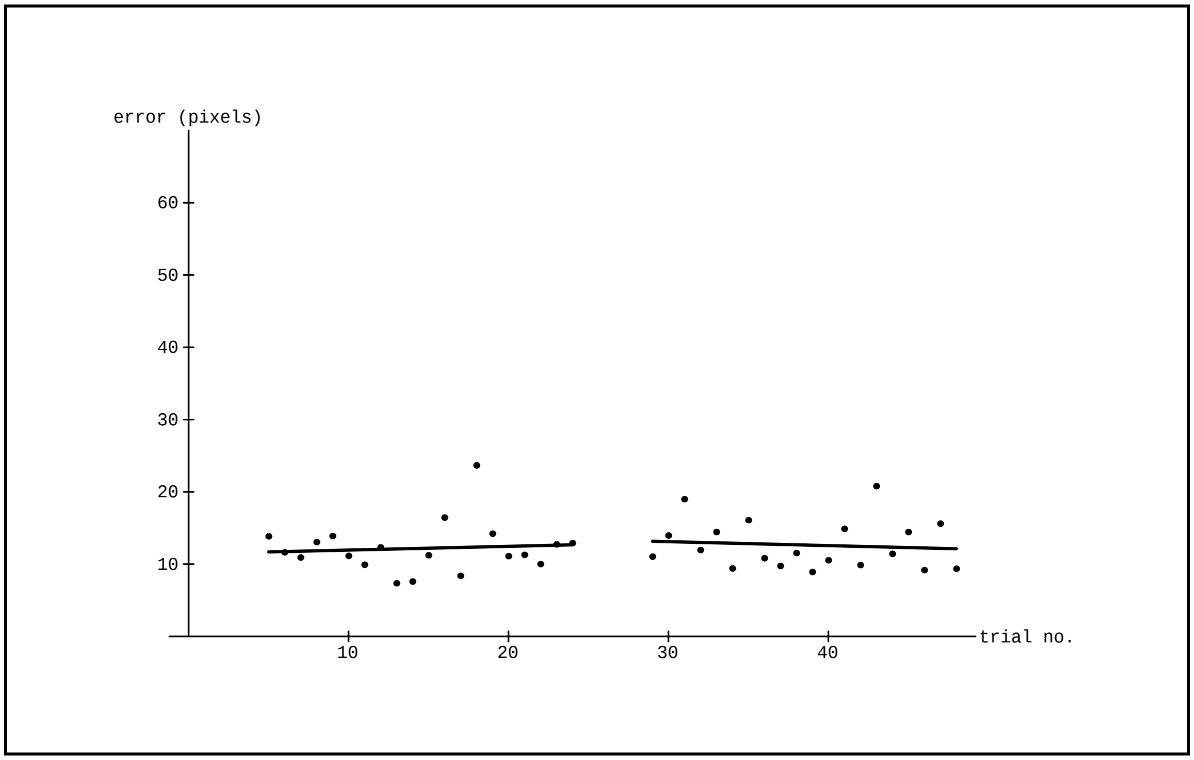


Fig 4.11 Mean trial errors for direct manipulation (DM1: trials 5-24 and DM2: trials 29-48).

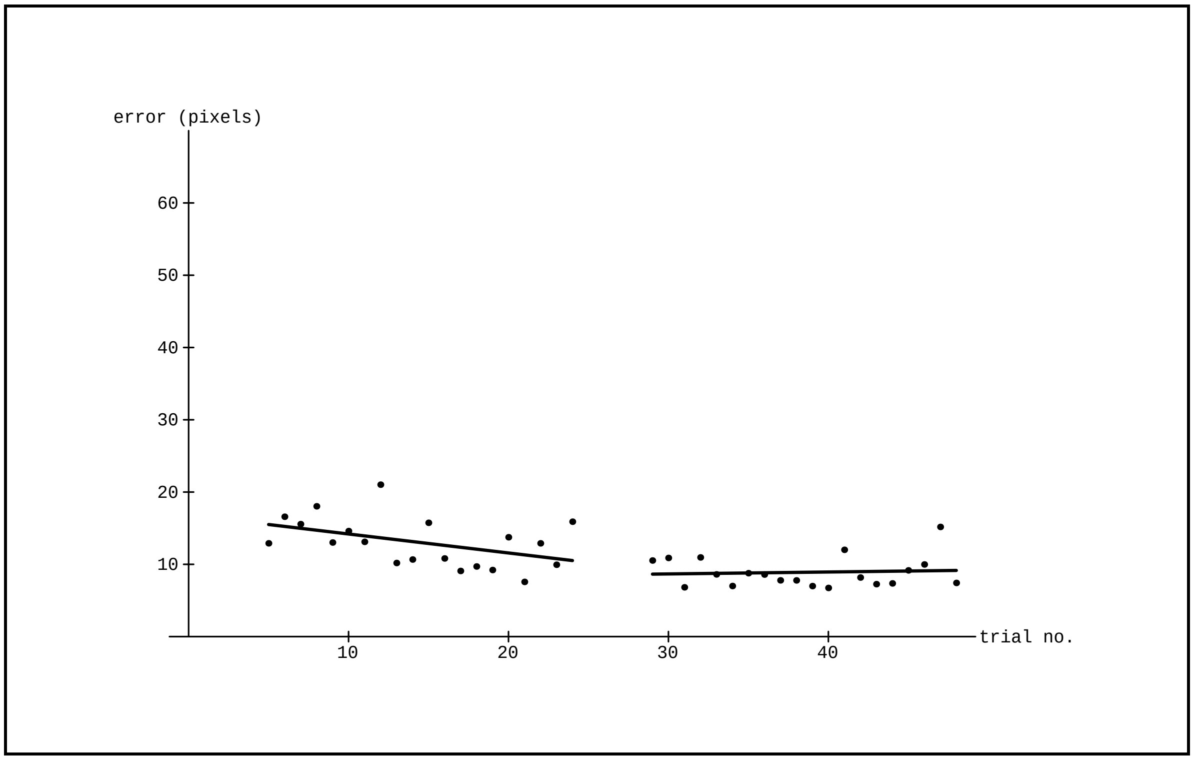


Fig 4.12 Median trial errors for control vertex manipulation (CV1: trials 5-24 and trials 29-48).

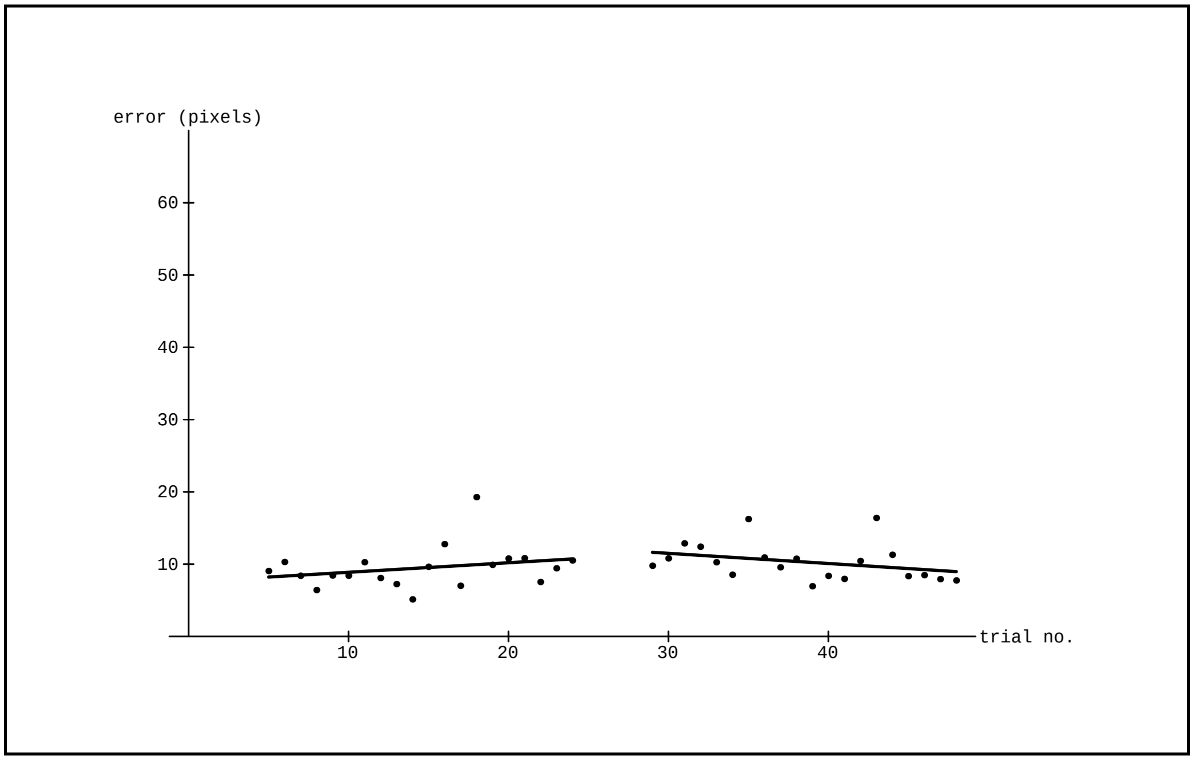


Fig 4.13 Median trial errors for direct manipulation (DM1: trials 5-24 and DM2: trials 29-48).

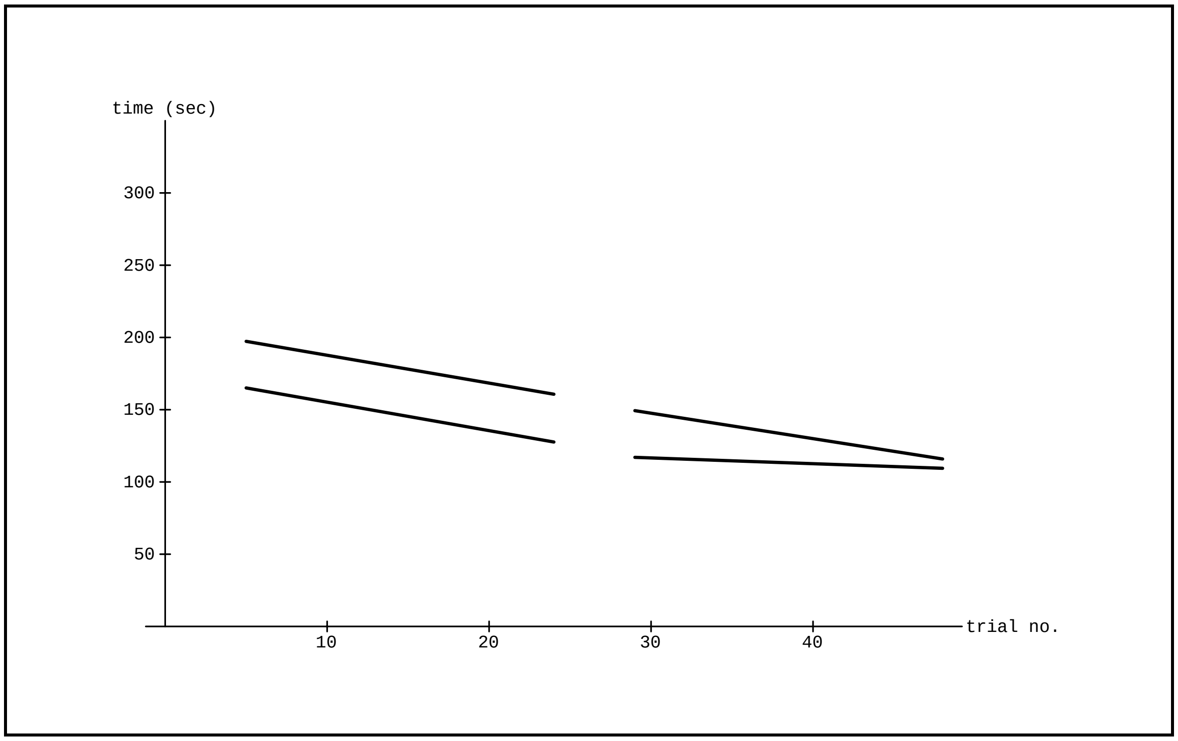


Fig 4.14 Regression lines for mean trial times (CV1: y = 207 − 1.93x ; CV2: y = 200 − 1.76x ; DM1: y = 1.75 − 1.97x ; DM2: y = 129 − 0.40x).

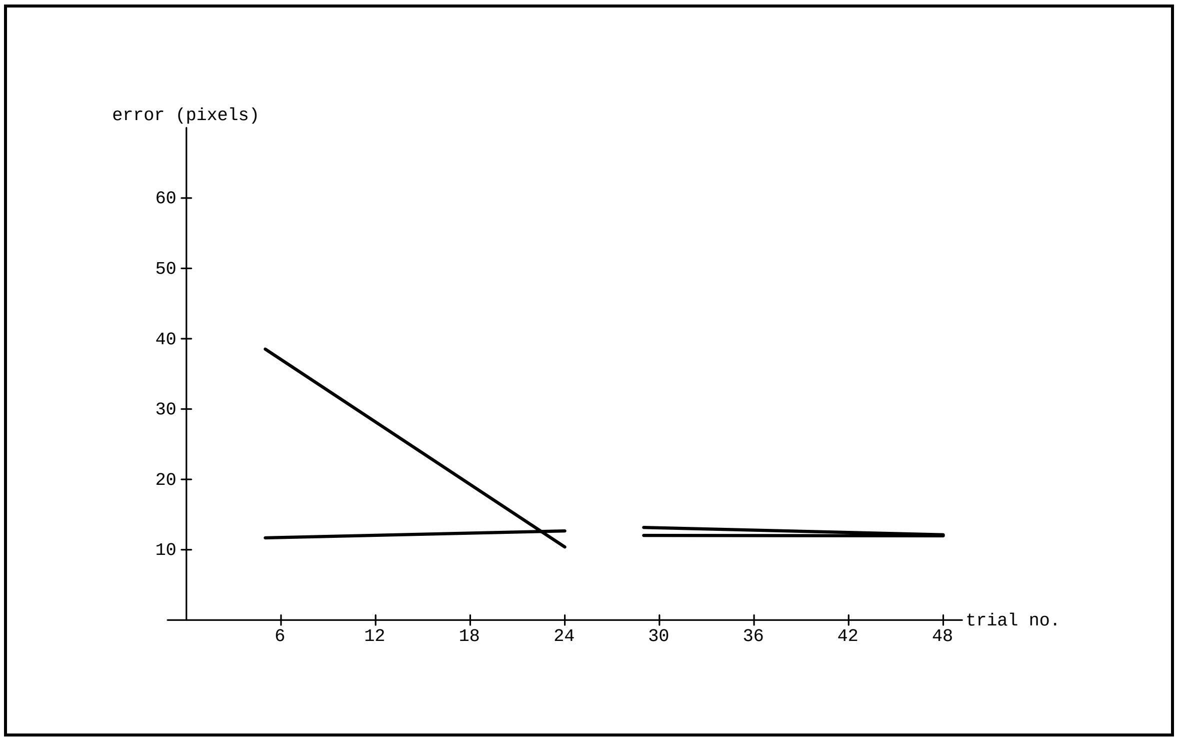


Fig 4.15 Regression lines for mean trial errors (CV1: y = 46 − 1.480x ; CV2: y = 12 − 0.003x ; DM1: y = 11 + 0.052x ; DM2: y = 15 − 0.055x).

Chapter 5

A Testbed for 3D Interaction Experiments

The software used for the direct manipulation experiment is actually a subset of a larger program with an abundant list of options including depth cues and parameters to customize the 3D task. This source code is a re-working of Ruest's experiment software [Ruest89]. The modification of his software to work with 3D curves was fairly easy because the Silicon Graphics library routines (GL) works in a 3D world by default. However, the revision of the user interface to handle 3D curve-matching generated many questions to which there were many possible answers. For example, how should 3D movement of the control points be handled using the mouse as the input device? Instead of just offering one interface to the curve-matching task, a number of distinct interfaces were implemented. In doing so, their effectiveness for this particular 3D task could be tested first hand. Some of these interfaces were well-suited for the curve-matching task, while others appeared to be more effective for other 3D tasks.

Besides the different 3D interface options, the environment and the specific task have a number of options to change their properties. Many aspects of the program can be changed either from a pop-up menu, command-line arguments, or a parameter file, making the software quite versatile. For example, the thickness of the curves or the speed of the mouse can be changed from a menu rather than updating a constant in the source code and re-compiling. With such versatility, the software is very useful for the design of future experiments. The researcher can easily compare a variety of techniques and environments without rewriting parts of the source code. In fact, an objective comparison is possible because match evaluation routines and other software for running an actual experiment are already available. For the same reason, when the design of the experiment has been completed, preparing the software for the formal experiment requires minimal work.

5.1 The 3D Task

The testbed is equipped with other 3D tasks besides 3D curve-matching. After matching some 3D curves with the newly written program, there was some concern that matching 3D splines might not be a suitable task for an experiment investigating 3D interface design. As an alternative, a docking task was implemented. It involved moving and orienting an object in three dimensions to try to fit it into a hole in a surface that was cut to fit the object in only one orientation (similar to puzzles for children). The task was conceptually very simple, requiring little background knowledge, yet it involved manipulation of the object in all six degrees of freedom (three translational and three rotational).

After a few modifications to the 3D curve-matching task, it was decided that the task was adequate for the study of 3D interaction. The remainder of this chapter will discuss the options provided by the software. Some have been implemented specifically for the curve-matching task, but apply equally well to the docking task and other tasks.

5.2 Movement of Control Points

The technique used in the direct manipulation experiment was to provide unconstrained two dimensional movement on a plane parallel to the screen, identical to that used in the 2D curve-matching task. The third degree of motion of the control point was manipulated by rotating the curves and then moving it to a new location. This method is probably the most intuitive if using a 2D input device, and is also the technique used in most 3D CAD and modelling packages where the simultaneous display of multiple 2D projections is often provided.

A technique offering constrained 3D movement of the control points does allow manipulation of each of the three dimensions without the need for rotating the curves, as opposed to the previous technique. Depending on which of the three mouse buttons is used to select a control point, movement is either in the *x*, *y*, or *z* direction. Disadvantages of this technique are that unconstrained movement is not possible and two of the three mouse buttons execute one of two actions (moving a control point or rotating the curves/ending the trial), depending on the location of the mouse (on a control point or not on a control point). The multiple uses of these buttons make the interface more complicated than the previous technique.

A third technique provides both constrained and unconstrained movement. It uses only two buttons to handle the movement by recognizing double-clicking[4] and chording[5] as well as single button clicks. The right button serves just the one function of exiting the task. Tables 5.1, 5.2 and 5.3 map the mouse input to the actions that they perform.

In comparing these techniques, some issues need to be investigated. One issue is whether or not constrained movement is useful. For curve-matching, constrained movement does not seem to be useful unless an orthogonal projection is used with the following strategy: match the curves perfectly in the initial view, rotate the curves by ninety degrees, then match the curves in the third dimension by moving the control points either horizontally or vertically depending on the rotation. If the curves are matched in one view, then two of the three coordinates for each control point are correct. To complete the match, move each point such that only the last of the three coordinates is modified (i.e., constrained movement). This strategy appears to be optimal, because only two views have to be matched, and only one rotation is made. Without constrained movement, it is difficult to match the last coordinate without upsetting the match of the first two coordinates.

A second issue is whether movement in depth is beneficial as compared to rotating the object first and then moving in the desired direction. The usefulness of movement in depth depends on the effectiveness of the given depth cues, but in most cases, the accuracy of movement in depth is far worse than that in the horizontal or vertical direction. Virtually no CAD or modelling software allows movement of objects in depth.

[4]Double-clicking is accomplished by pressing a mouse button twice in close succession.

[5]Chording is accomplished by pressing two mouse buttons at the same time.

Table 5.1 Mouse input and the corresponding actions: unconstrained 2D movement.

Mouse Button	Cursor on control point	Cursor NOT on control point
left	2D movement	---
middle	rotation	rotation
right	exit	exit

Table 5.2 Mouse input and the corresponding actions: constrained 3D movement.

Mouse button	Cursor on control point	Cursor NOT on control point
left	vertical movement	---
middle	horizontal movement	rotation
right	depth movement	exit

Table 5.3 Mouse input and the corresponding actions: constrained and unconstrained 2D movement.

Mouse Button	Cursor on control point	Cursor NOT on control point
left	vertical movement	---
middle	horizontal movement	rotation
right	exit	exit
double click left	2D movement	---
double click middle	2D movement	---
chord left-middle	2D movement	---

5.3 Curve Rotation

Many techniques are available for rotating objects in 3D. In general, 3D object rotation with a mouse is accomplished using a virtual trackball algorithm. For more details on virtual trackballs and definitions of some of the terms that will be used in this section, refer to Chapter Two. The algorithm of Chen, et al. [Chen90] is implemented in the curve-matching software with *positional*, *directional*, and *rate-controlled* interfaces. The *positional* interface is the standard trackball interface used in most applications. The *directional* and *rate-controlled* interface are very useful for the visualization of 3D objects because they allow the user to study the 3D information from the rotating objects without having to continually slide and pickup the mouse. On the other hand, they are not as effective as the *positional* interface for rotating an object during a construction task because precise rotations are more difficult.

The rotation technique used in the experiment was not a virtual trackball, but was a constrained technique allowing only ninety degree rotations about the *x* and *y* axes. This technique seemed to provide enough freedom for the curve-matching task. It was felt that providing a virtual trackball interface might complicate the task and jeopardize the results of the main factor studied, the manipulation technique. Given the greater freedom of a virtual trackball, subjects would likely spend too much time rotating the curves rather than matching them.

The last alternative available in the curve-matching software is to provide multiple orthogonal views instead of any rotation technique at all. Similar to most 3D modelling software, three small windows can be laid out on the display showing the front, top, and side view of the curves. Each window is essentially a 2D curve-matching task. In informal studies, this interface turned out to be inferior to a large, single window with a rotation technique. For complicated curves, it seemed nearly impossible to visually merge the information from the three windows to obtain a complete mental picture of a 3D curve. As mentioned in Chapter Two, the smooth rotation of an object is a very good depth cue; this is available with any of the virtual trackball techniques.

5.4 Depth Cues

A comprehensive list of depth cues was given in Chapter Two. A subset of these was implemented in the curve-matching software. The two display techniques for the curves are perspective and stereo perspective. The full set of parameters for the two projections are modifiable from pop-up menus during the curve-matching task. These parameters include the viewer distance from the monitor for perspective and the interocular distance for stereo.

Gouraud shading can be used both in the curves and in the background as an optional depth cue. For the curves, the colour can be interpolated between two boundary colours with the weighting dependant on the depth of that part of the curve. This technique is sometimes referred to as z axis modulation. Usually, the boundary colours are chosen so that the curves approach the background in colour the farther away it is. The background can be shaded to simulate a single light source behind the subject by interpolating the colours of the walls to be darker, the farther they are from the subject.

Some preliminary work with shadows has been implemented. Shadows of the control points can be orthogonally projected onto the left, right, top, and bottom walls. These shadows give the subject a good idea of the depths of the control points. Further work will involve shadows for the curves and the ability to manipulate the control points by selecting the shadows. This will provide another method for moving the points in three space using a mouse.

5.5 Spline Curves

There are a number of options dealing specifically with the curves. The spline formulation to use may be chosen from the following: B-spline, rational B-spline, beta2 spline, and natural spline. The number of control vertices for the curve may also be set. For B-splines, the formulation used in the direct manipulation experiment, either control vertex or direct manipulation may be selected. The number of end control vertices on the controlled curve that are fixed to the corresponding vertices on the target curve may also be set. In the experiment, three vertices on either end of the curves were fixed. The width of the curves as represented on the display can be modified from the pop-up menu. Finally, display of the solution (i.e., the control vertices on the target curve) may be toggled on and off. This option is used for debugging the software, but not in actual experiment trials.

5.6 Other Options

Most of the options for the curve-matching software are selected from a parameter file, the command-line, or a pop-up menu. A very important option is whether or not the trial is to be run in experiment mode. In this mode, the pop-up menu is disabled and a special quit menu is enabled whose only entries are "quit" and "resume".

Additional miscellaneous options are listed below.

- The speed or sensitivity of the mouse can be modified from the pop-up menu.
- The colours for the background walls can be changed from the menu.
- Control points may be represented as wire frame or a solid cube.
- Any window or icon appearing during the trial or tutorial may be moved or re-sized.

Most of the options discussed in this chapter may be invoked when running the testbed from pop-up menus, parameter files or command-line, and may be used in a formal experiment. Not all of the options can be implemented without editing the source code. This is a topic for future work.

Chapter 6

Summary and Future Work

Three major contributions were made in this research. First, a prototype testbed for the study of 3D interaction techniques was written. Using this software, one of a number of simple tasks can be performed using some combination of these interaction techniques. Second, additional experience was gained in the design of a 3D curve-matching experiment using experiment software similar to that used in previous 2D work. Finally, an experiment was conducted to compare direct manipulation of B-spline curves with control vertex manipulation of B-spline curves. The results provided evidence that direct manipulation was the better technique.

This experiment was the first of the curve-matching experiments dealing with 3D splines. At the same time, the attention has shifted towards the 3D interaction techniques. Further research in this area includes both enhancements to the testbed software and additional experiments. The software can be expanded to use other input devices and run different 3D tasks. It can also be generalized to simplify the implementation of such modifications. The list of future experiments to be performed is extensive, involving both curve-matching and other tasks. Some of the more immediate proposed studies are listed below:

- Running a similar experiment with “shadows” of the control points and curves orthogonally projected onto each of the four side walls of the box enclosing the curves. Besides providing depth information, the shadows of the control points can be picked and dragged along the wall, forcing the actual control point to move in response. Using this interface, the subject could conceivably complete the entire match in 3D without changing the orientation of the curves. Essentially, five views would be displayed on the screen at all times, four of which would be at reduced resolution as shadows.

- Representing the curves as smooth, generalized cylinders and applying specular reflection to them. Specular reflection should provide a very good curvature cue.

- Using the ADL-1 Head Tracker to change the view of the curves. Changing the view with head movement should be more natural than changing the orientation with the mouse. The subject’s hand that is used to control the mouse is relieved of a task, leaving it with the lone job of manipulating control points, possibly reducing some confusion. Unfortunately, the head tracker does not allow more than about forty-five degrees of rotation in any direction. It is not known whether this amount is adequate for matching the curves in depth. The following experiment should be conducted as a preliminary study.

- Investigate how the amount of rotation allowed by the ADL-1 Head Tracker affects the accuracy in depth of a simple task. This task could be to place a point at the mid-point of a line segment (not actually drawn) connecting two fixed points, or to place a point at the center of a cube. In this experiment, subjects would perform the task given varying amounts of rotation. As an initial prediction, the amount of rotation allowed should correlate positively with the accuracy in depth.

- Using the Spaceball to change the orientation of curves. As with the head tracker, the mouse is freed of the task of changing views. Instead, the subject would use his left hand to control the Spaceball. Capable of three degrees of rotation, the Spaceball should be very intuitive to use for this application.

All of these extensions should be easy to include in the existing testbed, although it is anticipated that a re-design of the testbed will at some point be required after more experience is gained with it.

Bibliography

[Bart87] Bartels, Richard H., John C. Beatty, and Brian A. Barsky, *An Introduction to Splines for Use in Computer Graphics & Geometric Modelling*, Morgan Kaufmann, Inc., 1987.

[Bart89] Bartels, Richard H., and John C. Beatty, “A Technique for the Direct Manipulation of Spline Curves”, *Graphics Interface Conference Proceedings*, 1989, p. 33-39.

[Bart93] Bartels, Richard H., John C. Beatty, Kellogg S. Booth, Eric G. Bosch, and Pierre Jolicoeur, “Experimental Comparison of Spline Bases Using the Shape-Matching Paradigm”, *ACM Transactions on Graphics*, 1993. To appear.

[Bosch87] Bosch, Eric G., “Workstation-Based Shape Matching Experiments”, Masters Thesis, University of Waterloo, 1987.

[Chen88] Chen, Michael, S.J. Mountford, Abigail Sellen, “A Study in Interactive 3-D Rotation Using 2-D Control Devices”, *Computer Graphics, SIGGRAPH Conference Proceedings*, 1988, Vol. 22, Num. 4, p. 121-129.

[Devo82] Devore, Jay L., *Probability and Statistics for Engineering and the Sciences*, Brooks/Cole, 1982.

[Foley90] Foley, James D., Andries van Dam, Steven K. Feiner, John F. Hughes, *Computer Graphics Principles and Practice*, 2nd edition, Addison Wesley, 1990.

[Forr86] Forrest, A.R., “User Interfaces for Three-Dimensional Geometric Modelling”, *Proceedings 1986 Workshop on Interactive 3D Graphics*, 1986, p. 237-249.

[Fors90] Forsey, David R., *Motion Control and Surface Modeling of Articulated Figures in Computer Animation*, PhD Thesis, University of Waterloo, 1990.

[Glass72] Glass, G.V., P.D. Peckham, and J.R. Sanders, “Consequences of Failure to Meet

Assumptions Underlying the Fixed Effects Analysis of Variance and Covariance”, *Review of Educational Research*, 1972, Vol. 42, p. 237-288.

[Hodg89] Hodges, Larry F., and David F. McAllister, “Computing Stereographic Views”, *ACM SIGGRAPH Course Notes*, 1989, Num. 24, p. 4.1-4.30.

[John89] Johnson, Phil, and Richard DeHoff, “Field Sequential Stereoscopic Graphics Systems and Applications”, *ACM SIGGRAPH Course Notes*, 1989, Num. 24, p. 3.1-3.7.

[Newm79] Newman, William M., and Robert F. Sproull, *Principles of Interactive Computer Graphics*, 2nd edition, McGraw-Hill, Inc., 1979.

[Ruest89] Ruest, Paul, “An evaluation of tension within an extensible spline testing facility”, Masters Thesis, University of Waterloo, 1989.

[Ruest] Ruest, Paul, Richard H. Bartels, John C. Beatty, and Pierre Jolicoeur, “An Experimental Evaluation of Tension Parameters in Cubic Beta-Splines and Rational B-Splines”, *IEEE Computer Graphics and Applications*. To appear.

[Shne83] Shneiderman, Ben, “Direct Manipulation: A Step Beyond Programming Languages”, *IEEE Computer*, 1983, Vol. 16, Num. 8, p. 57-69.

[Shne87] Shneiderman, Ben, *Designing the User Interface*, Addison-Wesley, Inc., 1987.

[Shoe92] Shoemake, Ken, “ARCBALL: A user interface for specifying three-dimensional orientation using a mouse”, *Graphics Interface Conference Proceedings*, 1992, p.32-37.

[Wang92] Wanger, Leonard R., “The Effect of Shadow quality on the Perception of Spatial Relationships in Computer Generated Imagery”, *ACM Symposium on Interactive 3D Graphics*, 1992, Vol. 25, Num. 2, p. 39-42.

Appendix A

Help Text Pages

The seven page tutorial used in the experiment is listed on the following pages along with each demo window that the subject sees. Some pages are repeated because they contain multiple demos. For these pages, new material is shown in boldface. In the actual tutorial, new material is shown in yellow as opposed to white, the colour used for text displayed previously on the screen.

- Page 1 -

University of British Columbia
Computer Graphics Curve-Matching Experiments

The experiment you are about to participate in is part of an investigation into the effectiveness of TWO curve drawing techniques for manipulating three-dimensional (3D) curves. During the experiment, you will be matching curves in 3D using each drawing technique.

Before you begin the actual experiment, you will be led through a tutorial that will familiarize you with the computer graphics equipment and the curve drawing process.

Please take your time during the tutorial. It is important that you understand how the computer graphics equipment works. If at any time during the tutorial or during the actual experiment you are uncertain as to what is required of you, please feel free to ask the experiment supervisor for assistance.

The primary piece of graphics equipment that you will be using is called a MOUSE and is positioned on a mouse pad in front of the graphics terminal. By moving the mouse over the surface of the mouse pad you can control the position of the red arrow, called a CURSOR, on the graphics display. Note that the movement of the mouse is not absolute, but is instead relative to its previous position while in contact with the pad. Thus you can lift the mouse off the mouse pad, place it at a new location on the pad, and then continue moving the cursor.

During this tutorial the mouse will be used to PICK certain objects on the graphics display. To pick an object, position the cursor on the object and then press the appropriate button (LEFT, MIDDLE, or RIGHT) on the top of the mouse.

The boxes visible on the left-hand side of the graphics display represent a table of contents for the introductory pages you are now reading. Each box corresponds to a different page of this introduction. There are only two boxes visible at the moment: as you read additional pages, additional boxes will become visible.

Note that the boxes are outlined in different colours. A red outline indicates that the corresponding page is currently being read, while a green outline indicates that the corresponding page has not yet been read. Once a page has been read it will be outlined in white. Look, for example, at the currently visible boxes: the first is outlined in red because it corresponds to this page (the one you are now reading); the second box is outlined in green because it has not yet been read.

When you are finished reading a page, you move on to another by using the mouse to pick a box from the column of visible boxes on the left of the graphics display. You can review previously read pages by picking one of the white-outlined boxes, or you can continue with new material by picking the green-outlined box. If you pick the box outlined in red, nothing will happen, since the page picked is the page currently displayed. So, to read the next page, pick the green outlined box labelled Page 2.

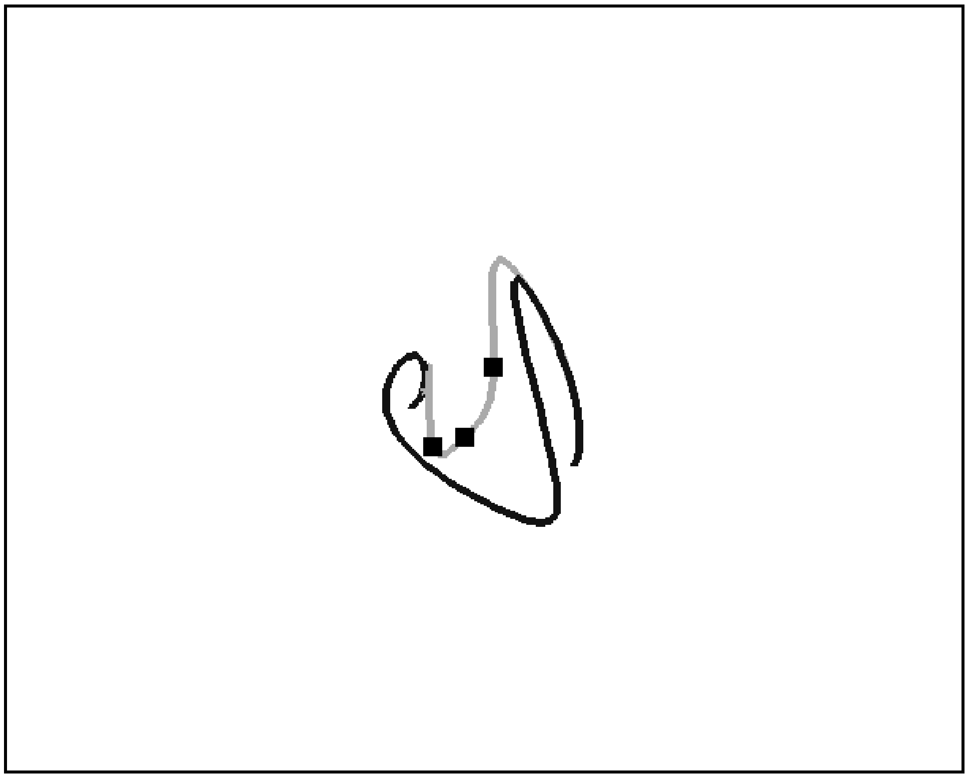
- Page 2 -

This experiment is divided into two sessions which are to be completed on different days. You will be performing one of the two sessions right now. You will be using a different curve drawing technique in each session.

Each session comprises 24 trials followed by a series of subjective ratings (which is discussed on the last page). They are separated into 4 initial practice trials and 20 recorded trials. The number of trials remaining will be displayed in the lower-left corner of the graphics display.

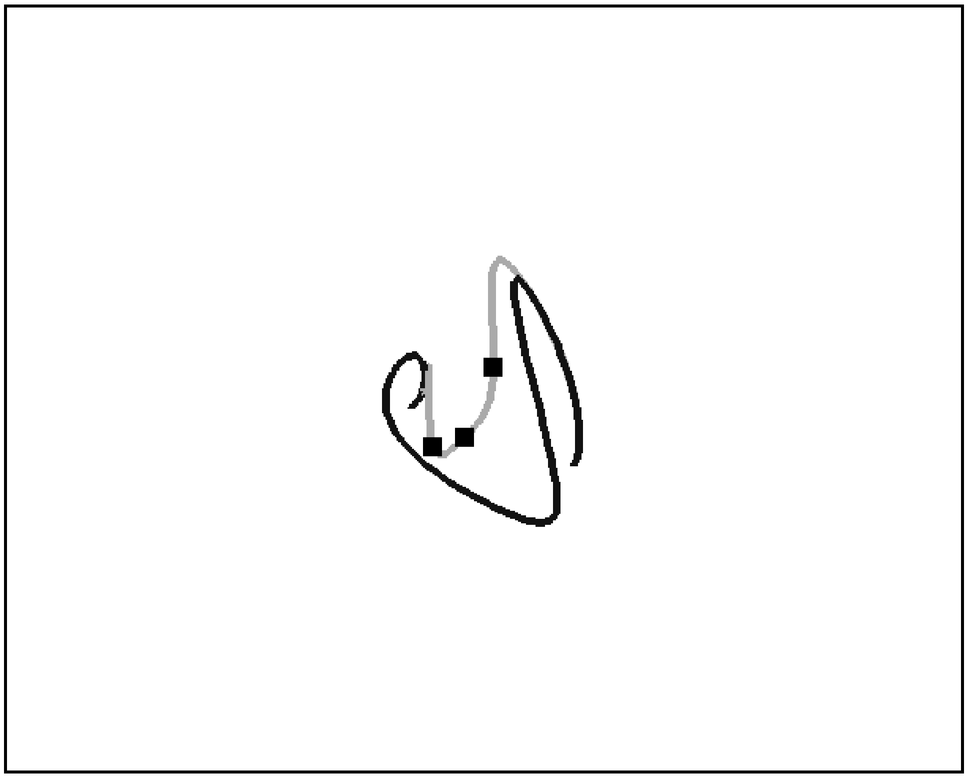
During each trial you will be presented with two 3D curves drawn on the graphics display (see below). Your task is to manipulate the BLUE curve so that it matches the RED curve in all three dimensions. The BLUE curve can be manipulated by moving any of the three small BLACK boxes, representing the control points. Your goal during each trial is to match the curves as closely as possible in a short amount of time.

Notice that the endpoints of the two curves are already matched. You will only be manipulating the middle portion of the BLUE curve. Since the curves are three dimensional, you will have to ROTATE the curves and work with them from different views in order to match them.



```
                                                                - Page 3 -

The curves can be rotated by 90 degrees in any direction any number of times
by using the MIDDLE mouse button.  Pressing and holding down the middle button
causes a set of diagonal cross-hairs, a circle and possibly an arrow to be
displayed.  Try this now, when you are done, press the right button and select
"Finished matching, stop timer".
```



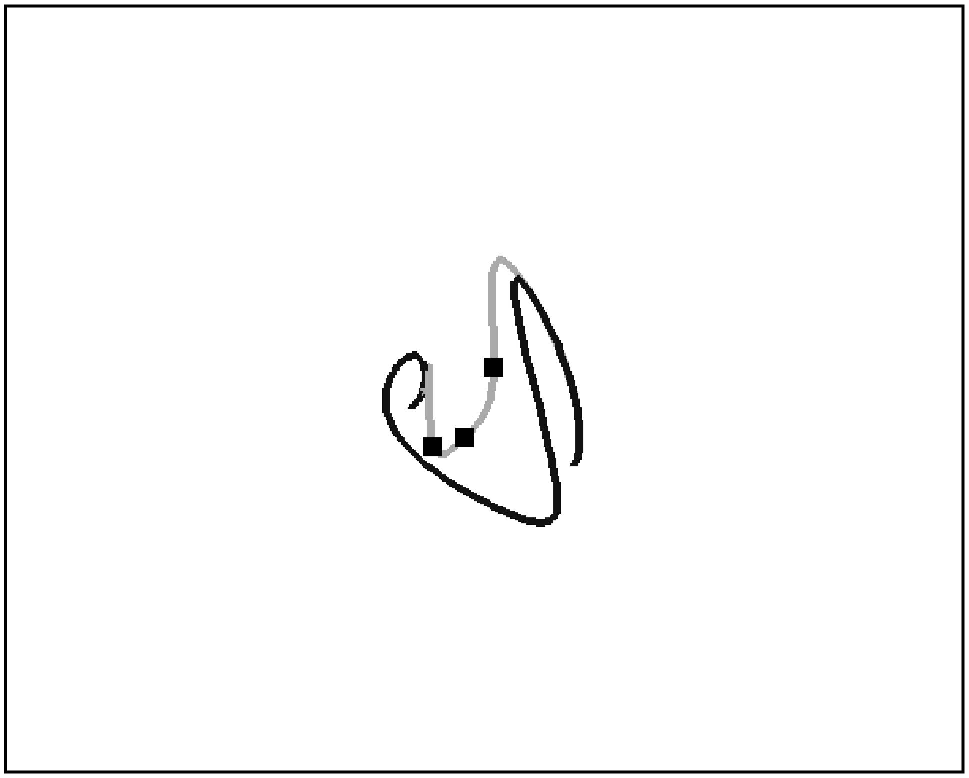
- Page 3 -

The curves can be rotated by 90 degrees in any direction any number of times by using the MIDDLE mouse button. Pressing and holding down the middle button causes a set of diagonal cross-hairs, a circle and possibly an arrow to be displayed. Try this now, when you are done, press the right button and select "Finished matching, stop timer".

The cross-hairs and circle essentially divide the screen into five regions (left, right, top, bottom, and center). Each region corresponds to a direction of rotation (the center region is for no rotation), so by moving the cursor to a particular region, you can rotate the curves in the desired direction. Rotation does not actually occur until the mouse button is released. The arrow lets you know in which direction the curves will rotate if you release the MIDDLE mouse button now. The following table summarizes the above points:

CURSOR REGION	**ARROW**	**ROTATION**
right	**right**	**to the right by 90 degrees**
top	**up**	**upwards by 90 degrees**
left	**left**	**to the left by 90 degrees**
bottom	**down**	**downwards by 90 degrees**
center	**none**	**none (i.e., cancel rotation)**

Try rotating the curves now, when you are done, use the right button to stop.

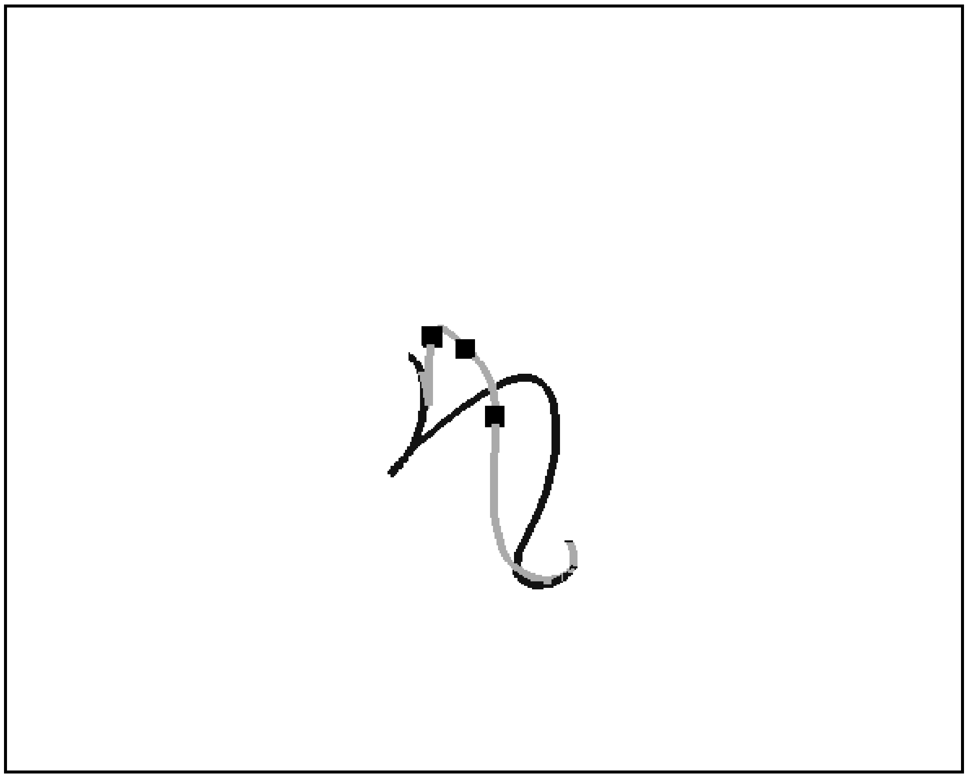


- Page 4 -

You can move a control point by positioning the cursor on the black box representing it, and then holding down the left mouse button while you move the mouse. Pressing the mouse button causes the red cursor and the control points to disappear, confirming your selection. When you release the mouse button, the red cursor and the control points reappear, indicating that you are no longer moving that control point.

As you move a control point, the BLUE curve changes shape in response to the new position of the control point. If you take a closer look at the two curves, you’ll notice that the ends have already been matched.

Try using the cursor to move a few control points now and when you are finished, press and hold down the RIGHT mouse button and select "Finished matching, stop timer".



- Page 4 -

You can move a control point by positioning the cursor on the black box representing it, and then holding down the left mouse button while you move the mouse. Pressing the mouse button causes the red cursor and the control points to disappear, confirming your selection. When you release the mouse button, the red cursor and the control points reappear, indicating that you are no longer moving that control point.

As you move a control point, the BLUE curve changes shape in response to the new position of the control point. If you take a closer look at the two curves, you’ll notice that the ends have already been matched.

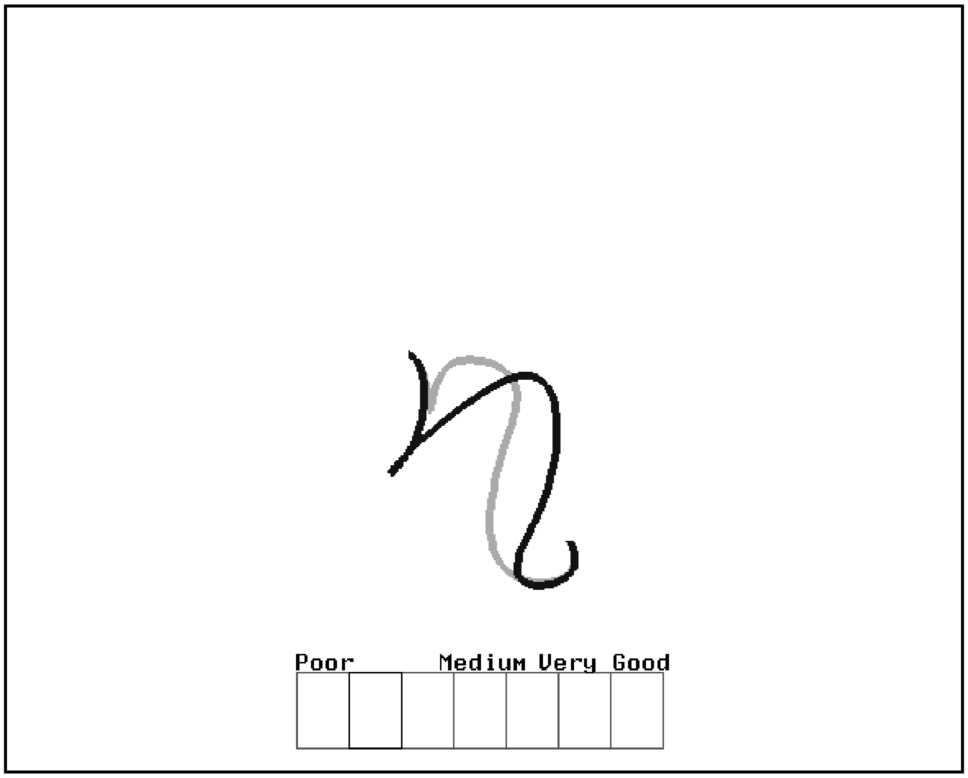
Try using the cursor to move a few control points now and when you are finished, press and hold down the RIGHT mouse button and select "Finished matching, stop timer".

Remember that you have to match the curves in 3D, so you will be constantly moving control points and rotating the curves. You know that you have successfully matched the curves in 3D when EVERY view of the curves is matched.

- Page 5 -

At the end of every trial you will be asked to rate the quality of your match. The rating scale has seven values, from POOR through MEDIUM to VERY GOOD. It is suggested that during the course of the experiment you make use of the entire scale by tailoring the rating range to your own personal performance, so that your best match is VERY GOOD and your worst is POOR.

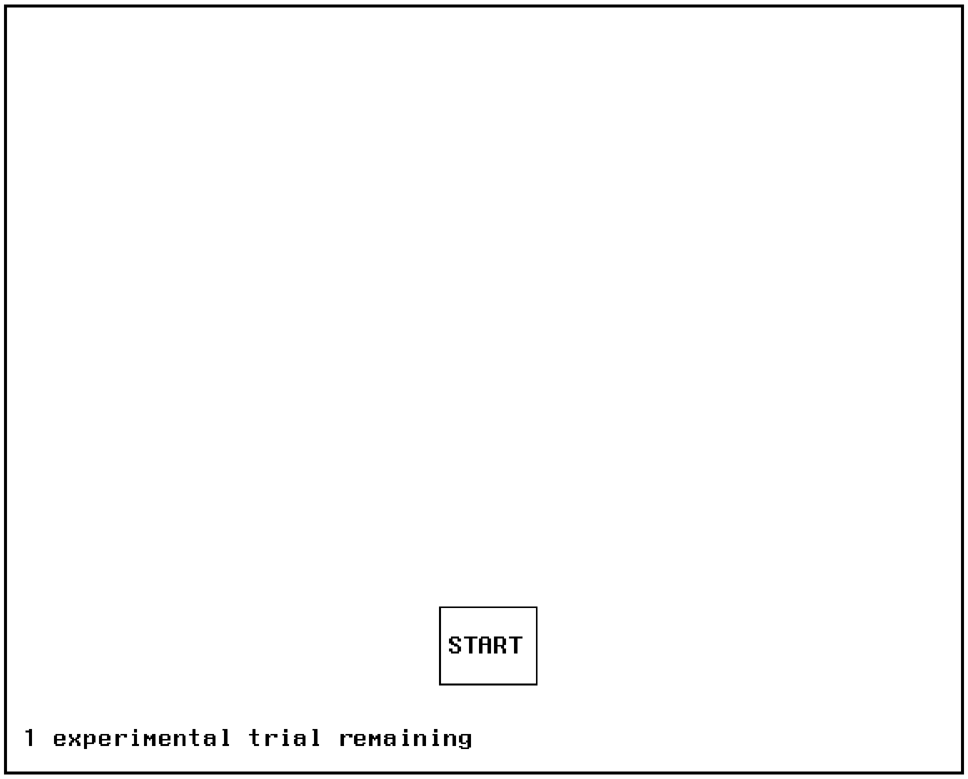
To make a selection on the rating scale, move the mouse so that the cursor is on the appropriate box. As you move the cursor along the scale, the currently selected box is temporarily highlighted. When you are satisfied with your selection, press any mouse button while the desired box is selected to record your rating. Try this now.



- Page 6 -

By now you should be familiar with the mouse and how it is used to pick objects.

During a trial there are three screen configurations: the START screen, the INTERACTION screen, and the RATING screen. In the starting configuration you are shown an empty window and a box labelled START. When you are ready to start a trial, pick the box labelled START. Try this now.

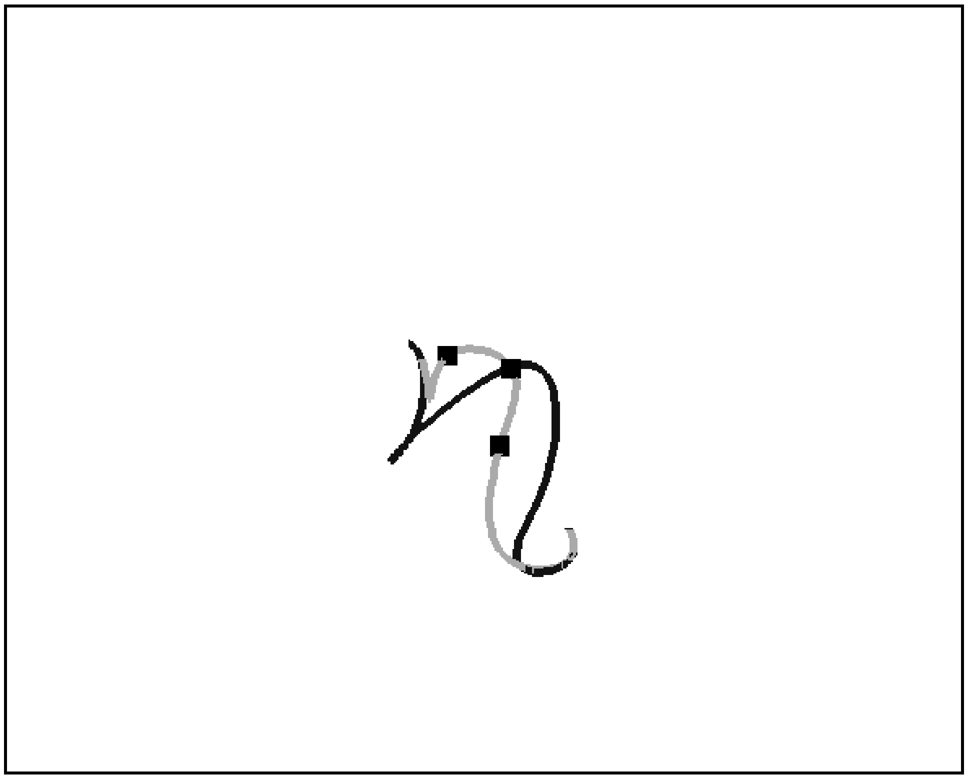


- Page 6 -

By now you should be familiar with the mouse and how it is used to pick objects.

During a trial there are three screen configurations: the START screen, the INTERACTION screen, and the RATING screen. In the starting configuration you are shown an empty window and a box labelled START. When you are ready to start a trial, pick the box labelled START. Try this now.

**The display then changes to the interaction screen. During this stage you are to manipulate the controlled curve so that it matches the target curve as closely as possible. Remember that these curves are three dimensional and require matching in EVERY view. When you are satisfied with your match, or decide that you cannot improve it, press and hold down the right mouse button and select "Finished matching, stop timer".
Try matching the curve now.**



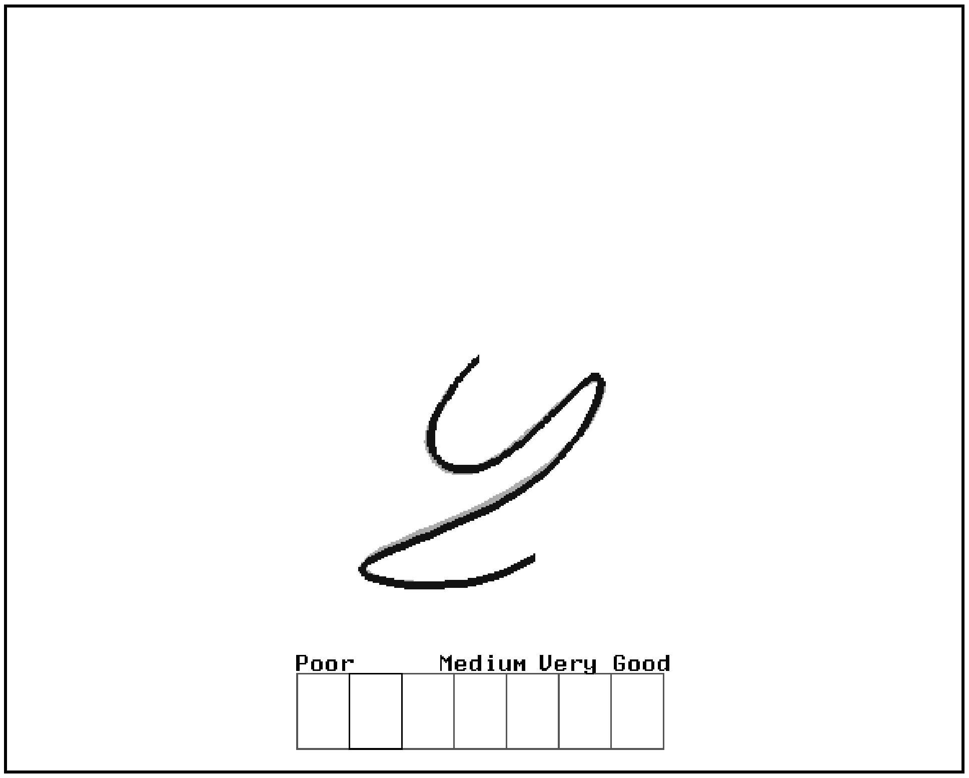
- Page 6 -

By now you should be familiar with the mouse and how it is used to pick objects.

During a trial there are three screen configurations: the START screen, the INTERACTION screen, and the RATING screen. In the starting configuration you are shown an empty window and a box labelled START. When you are ready to start a trial, pick the box labelled START. Try this now.

The display then changes to the interaction screen. During this stage you are to manipulate the controlled curve so that it matches the target curve as closely as possible. Remember that these curves are three dimensional and require matching in EVERY view. When you are satisfied with your match, or decide that you cannot improve it, press and hold down the right mouse button and select "Finished matching, stop timer".
Try matching the curve now.

The rating scale is now superimposed on the interaction screen. Make your rating selection by moving the cursor within the appropriate box and then pressing any mouse button. Try this now.



- Page 6 -

By now you should be familiar with the mouse and how it is used to pick objects.

During a trial there are three screen configurations: the START screen, the INTERACTION screen, and the RATING screen. In the starting configuration you are shown an empty window and a box labelled START. When you are ready to start a trial, pick the box labelled START. Try this now.

The display then changes to the interaction screen. During this stage you are to manipulate the controlled curve so that it matches the target curve as closely as possible. Remember that these curves are three dimensional and require matching in EVERY view. When you are satisfied with your match, or decide that you cannot improve it, press and hold down the right mouse button and select "Finished matching, stop timer".
Try matching the curve now.

The rating scale is now superimposed on the interaction screen. Make your rating selection by moving the cursor within the appropriate box and then pressing any mouse button. Try this now.

One important thing to note is that it is not possible to go back to the interaction screen once you have selected the rating screen, and similarly it is not possible to go back to the start screen once you have selected the interaction screen.

- Page 7 -

Congratulations! You have just completed one entire practice trial. If you do not feel prepared to begin the experiment now, please go back and review the necessary instruction page(s).

It is important that you complete all of the 24 trials in one uninterrupted sitting. If you wish to pause during a session, please do so between trials (i.e., when the START button is displayed), since you are being timed.

After the 24 trials have been completed, you will be given 20 pre-matched trials. All that you have to do is study the matches by rotating the curves through the different views, and then rate them in the same way that you have rated your own matches. As in the regular trials, the MIDDLE mouse button rotates the curves and the RIGHT button brings up the rating scale.

Here is a SUMMARY of key points:

- Pressing the LEFT mouse button inside any control point will move the control point, reshaping the curve. (page 4)
- Pressing the MIDDLE mouse button rotates the curves. (page 3)
- Pressing the RIGHT mouse button pops up a menu to stop the trial.
- Since the curves are 3D, every view must be matched. (page 4)
- Match as quickly and as closely as you can. (page 2)

Please ensure that all these points are clear, reviewing the indicated page(s) if necessary.

Appendix B

Experiment Subject Forms

The following forms were filled out by each subject who took part in the experiment. The *Subject Information Form* and the *Consent Form* were completed before the experiment. The first page of the *Comments Form* was completed after the first session; the second page was completed after the second session.

Spline-based Curve Matching Experiments

Subject Information Form

Please provide the information requested below. This information will be held in strict confidence by the researchers.

Name: ______________________________

ID (if UBC student, use student ID):______________________________

Sex: ______________________________

Current Age: ______________________________

Are you primarily left or right handed? ______________________________

If you are a student, what department are you in and what year have you just completed:

____

Briefly describe your computer background (i.e., PC, Macintosh, etc.):

____

____

____

Have you ever had any experience with a mouse? ______________________________
If yes, please elaborate: ______________________________

____

Have you ever had any experience with a curve drawing program? ______________________________
If yes, please elaborate: ______________________________

____

Have you ever had any experience with spline mathematics? ______________________________
If yes, please elaborate: ______________________________

____

Spline-based Curve Matching Experiments

Consent Form

I agree to participate in the study entitled *Spline-based Curve Matching Experiments* being conducted by the Imager Laboratory of the Department of Computer Science at the University of British Columbia.

I understand that the data gathered by the computer program will only be seen by the researchers. The results which will be summarized in the experiment supervisor's Master's thesis will be stripped of all identifying codes.

I understand that my participation is voluntary and that I may withdraw from the study at any time.

Signature: ______________________________

Date: ______________________________

Spline-based Curve Matching Experiments

Comments Form

PLEASE COMPLETE THIS PAGE AFTER COMPLETION OF THE FIRST SESSION.

ID (if UBC student, use student ID):______________________________

What specific strategy did you use in moving control points to achieve a match? ________

____

____

____

____

What criteria did you use in rating the quality of your match? (i.e., accuracy of match, time taken, difficulty, etc.)

____

____

____

____

Any problems or comments on the tutorial? ______________________

____

____

____

Any other comments on the first session? ______________________

____

____

____

____

THANK YOU, PLEASE VERIFY THE TIME AND DATE FOR THE SECOND SESSION.

PLEASE COMPLETE THIS PAGE AFTER COMPLETION OF THE SECOND SESSION.

What specific strategy did you use in moving control points to achieve a match?

What criteria did you use in rating the quality of your match? (i.e., accuracy of match, time taken, difficulty, same as previous session, etc.)

In which session did you find the control of the curves easiest?

In which session did you think that you were the most successful in your final matches?

Which session did you enjoy the most?

Any other comments on the second session or on the experiment in general?

THANK YOU FOR TAKING PART IN THIS EXPERIMENT.

Appendix C

Background of Experiment Subjects

Table C.1 Background of Experiment Subjects

Subject	Sex	Age	L/R Handed	Background	Experience with			
					Comp-uter	Mouse	Curve Editor	Spline Math
1	F	25	R	Chemical Science	Y	Y	N	N
2	M	22	R	Metal &Materials Engineering	Y	Y	N	N
3	F	24	R	Education	Y	Y	N	N
4	M	22	R	Electrical Engineering	Y	YY	Y	Y
5	M	23	R	Civil Engineering	Y	YY	Y	N
6	M	29	R	Astronomy	Y	YY	Y	Y
7	M	25	R	Engineering Physics	Y	YY	N	N
8	M	32	R	Chemical Engineering	Y	Y	N	Y
9	M	24	R	Kinesiology	Y	Y	Y	N
10	M	22	R	Metal & Materials Engineering	Y	Y	N	N
11	M	29	R	Electrical Engineering	Y	Y	Y	Y
12	F	28	R	Metal & Materials Engineering	Y	N	N	N
13	M	31	R	Biology	Y	Y	N	N
14	M	28	R	Civil Engineering	Y	Y	Y	N
15	F	21	R	Metal & Materials Engineering	Y	Y	N	N
16	M	23	R	Engineering Physics	Y	Y	Y	N

YY - Very Experienced

Appendix D

Experiment Results

Results are given for the trial analysis (median of subjects' score for each trial) and subject analysis (median of all trials for a subject). They are divided into the following blocks: CV1 - control vertex manipulation for subjects' first session, CV2 - control vertex manipulation for subjects' second session, DM1 - direct manipulation for subjects' first session, and DM2 - direct manipulation for subjects' second session.

Table D.1 Trial Data - CV1

Trial	Median Time	Median Error	Mean Rating
1	169.23	12.91	5.125
2	175.35	16.58	5.125
3	167.92	15.56	5.625
4	173.57	18.03	5.125
5	138.51	13.02	5.250
6	185.02	14.60	5.250
7	198.56	13.11	5.250
8	136.03	21.02	4.875
9	213.04	10.18	6.250
10	160.10	10.66	5.875
11	128.60	15.75	4.875
12	144.70	10.81	5.125
13	164.28	9.08	5.125
14	149.62	9.70	5.500
15	156.96	9.21	5.500
16	160.94	13.75	5.375
17	138.12	7.57	5.875
18	130.37	12.91	6.000
19	158.15	9.94	5.375
20	173.27	15.90	5.625

Table D.2 Trial Data - CV2

Trial	Median Time	Median Error	Mean Rating
1	148.23	10.53	5.250
2	115.52	10.88	5.375
3	124.83	6.82	5.875
4	150.42	10.96	5.125
5	124.86	8.61	4.875
6	155.95	7.00	5.250
7	92.62	8.77	5.625
8	101.13	8.59	5.250
9	106.68	7.78	5.750
10	124.84	7.78	5.500
11	134.59	6.99	5.375
12	129.20	6.73	5.875
13	135.06	12.01	5.625
14	137.67	8.17	6.000
15	113.24	7.26	5.125
16	115.81	7.36	5.750
17	105.80	9.16	4.875
18	115.22	9.98	5.625
19	129.86	15.17	5.500
20	143.23	7.43	5.375

Table D.3 Trial Data - DM1

Trial	Median Time	Median Error	Mean Rating
1	137.15	9.05	4.875
2	157.91	10.30	5.375
3	168.15	8.39	5.375
4	155.86	6.42	5.250
5	149.07	8.43	5.625
6	139.54	8.40	5.375
7	136.92	10.27	5.250
8	154.66	8.08	5.375
9	120.20	7.24	5.375
10	173.39	5.12	5.875
11	135.10	9.63	5.000
12	126.06	12.77	5.375
13	140.46	7.01	5.500
14	190.80	19.27	4.500
15	113.77	9.91	5.500
16	133.71	10.78	5.375
17	105.59	10.82	5.500
18	115.62	7.53	5.500
19	127.61	9.43	4.750
20	127.20	10.51	4.875

Table D.4 Trial Data - DM2

Trial	Median Time	Median Error	Mean Rating
1	137.05	9.78	5.750
2	103.00	10.80	5.625
3	129.53	12.87	6.000
4	109.77	12.41	6.000
5	95.34	10.27	6.250
6	109.17	8.53	5.625
7	103.69	16.24	5.625
8	122.14	10.92	6.000
9	111.25	9.56	5.375
10	93.66	10.75	5.875
11	96.29	6.95	5.875
12	115.22	8.37	5.500
13	109.60	7.96	5.500
14	134.94	10.44	5.500
15	132.90	16.39	5.875
16	115.96	11.30	5.375
17	110.76	8.34	6.125
18	84.76	8.48	6.125
19	85.31	7.93	6.000
20	103.59	7.75	5.375

Table D.5 Subject Data - CV1

Subject	Median Time (sec)	Median Error (pixels)	Mean Rating
1	197.43	9.05	6.00
2	160.55	12.50	3.55
3	278.07	12.85	6.50
4	220.03	5.89	4.85
5	133.98	24.85	4.10
6	167.40	21.07	6.90
7	123.74	8.16	5.60
8	150.81	17.98	5.75

Table D.6 Subject Data - CV2

Subject	Median Time (sec)	Median Error (pixels)	Mean Rating
9	118.04	21.38	5.75
10	121.36	14.57	5.25
11	151.79	10.44	6.25
12	129.32	5.10	4.75
13	216.67	9.00	6.30
14	133.98	7.28	5.70
15	117.11	7.21	5.50
16	72.61	5.24	4.10

Table D.7 Subject Data - DM1

Subject	Median Time (sec)	Median Error (pixels)	Mean Rating
1	138.31	15.42	5.70
2	107.10	9.65	5.45
3	168.27	11.52	5.15
4	117.85	4.51	4.65
5	210.97	11.93	6.15
6	159.55	6.78	5.35
7	173.20	8.57	5.75
8	95.53	5.15	4.05

Table D.8 Subject Data - DM2

Subject	Median Time (sec)	Median Error (pixels)	Mean Rating
9	118.74	7.40	7.00
10	108.83	8.63	4.10
11	137.74	14.81	7.00
12	125.62	5.38	5.35
13	75.64	17.92	3.95
14	116.58	9.28	6.95
15	88.50	9.96	5.60
16	134.21	10.16	6.20

Appendix E

Description of Target Curves

There were twenty target curves used in the experiment, each one described by nine control vertices, P1 to P9.